ual: university of the arts london

| | |
|---|---|
| Title | Gestures in Machine Interaction |
| Type | Thesis |
| URL | https://ualresearchonline.arts.ac.uk/id/eprint/15579/ |
| Date | 2011 |
| Citation | Barker, Leon (2011) Gestures in Machine Interaction. PhD thesis, University of the Arts London. |
| Creators | Barker, Leon |

**Usage Guidelines**

Please refer to usage guidelines at http://ualresearchonline.arts.ac.uk/policies.html or alternatively contact ualresearchonline@arts.ac.uk.

License: Creative Commons Attribution Non-commercial No Derivatives

Unless otherwise stated, copyright owned by the author

# Gestures in Machine Interaction


Leon Barker

SCIRIA

University of the Arts London

A thesis submitted for the degree of

*Doctor of Philosophy*

August 2011

BRITISH LIBRARY

**IMAGING SERVICES NORTH**

Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

# BEST COPY AVAILABLE.

# VARIABLE PRINT QUALITY

BRITISH LIBRARY HSIRB

**IMAGING SERVICES NORTH**

Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

# TEXT BOUND CLOSE TO THE SPINE IN THE ORIGINAL THESIS

BRITISH LIBRARY

**IMAGING SERVICES NORTH**

Boston Spa, Wetherby
West Yorkshire, LS23 7BQ
www.bl.uk

# ORIGINAL COPY TIGHTLY BOUND

I would like to give a special thanks to Amy for being patient and compassionate and giving me the confidence to persevere. I would also like to thank Marlowe for giving me the spur and motivation to complete.

# Acknowledgements

I would like to thank my supervisors Dr Angela Geary from Northumbria University, Dr John Harrison from Imperial College London and Dr Jon Rimmer from City University London for their help and guidance. I would like to especially thank Dr Angela Geary who has been a great mentor, without her unwavering support I would not have reached this stage. Members of the SCIRIA research programme have also played an important support role and I appreciate their contribution to making my research fruitful and enjoyable. This work has been supported by a UAL Studentship, without which this research would not of been possible. Mum and Phil thanks for backing me up when it really counted, I cannot thank you enough. Thank you Ricardo for encouraging me to fulfil my dreams.

# Abstract

Unencumbered-gesture-interaction (UGI) describes the use of unrestricted gestures in machine interaction. The development of such technology will enable users to interact with machines and virtual environments by performing actions like grasping, pinching or waving without the need of peripherals. Advances in image-processing and pattern recognition make such interaction viable and in some applications more practical than current modes of keyboard, mouse and touch-screen interaction provide. UGI is emerging as a popular topic amongst Human-Computer Interaction (HCI), Computer-vision and gesture research; and is developing into a topic with potential to significantly impact the future of computer-interaction, robot-control and gaming. This thesis investigates whether an ergonomic model of UGI can be developed and implemented on consumer devices by considering some of the barriers currently preventing such a model of UGI from being widely adopted. This research aims to address the development of freehand gesture interfaces and accompanying syntax. Without the detailed consideration of the evolution of this field the development of un-ergonomic, inefficient interfaces capable of placing undue strain on interface users becomes more likely. In the course of this thesis some novel design and methodological assertions are made. The Gesture in Machine Interaction (GiMI) syntax model and the Gesture-Face Layer (GFL), developed in the course of this research, have been designed to facilitate ergonomic gesture interaction. The GiMI is an interface syntax model designed to enable cursor control, browser navigation commands and steering control for remote robots or vehicles. Through applying state-of-the-art image processing that facilitates three-dimensional (3D) recognition of human action, this

research investigates how interface syntax can incorporate the broadest range of human actions. By advancing our understanding of ergonomic gesture syntax, this research aims to assist future developers evaluate the efficiency of gesture interfaces, lexicons and syntax.

# Contents

# List of Figures

# List of Tables

# Nomenclature

### Roman Symbols

AR Augment reality describes the mixing of the real world and digital world into a single sensory space

Deictic Deitic gesture represent pointing actions

EGL Extended Gesture Lexicon a subset of the GiMI

EGL Extended Gesture Lexicon a subset of the GiMI

EMG Electromyography is the practice of inserting needle electrodes into the body in order to monitor the electrical charges released by the muscles

Epistemic Epistemic actions represent actions that provide haptic feedback gained from actions such as touching, feeling and squeezing

Ergotic Ergotic actions represent actions utilised in the physical manipulation of objects i.e twisting, pushing or pulling

GEf Gesture efficiency dataset compiled in chapter 4

GFL Gesture Face Layer prototypal gesture interface designed during this research

GiMI Gestures in machine interaction

mahalanbois distancing A statistical algorithm for calculating variance within data. Developed by Prasanta Chandra Mahalanobis.

PCA Principal Component Analysis

Posteriori A posteriori knowledge is proven through evidence and expresses an empirical fact unknowable through reason alone.

Priori A priori knowledge is an accurate assumption known independently of evidence or experience

SEMG Surface electromyography is the non invasive practice of monitoring the electrical charges released by the muscles, through the use of electrodes attached to the surface of the skin

Semiotic Semiotic actions represent actions used for communicating information

UGI Unencumbered gesture-interaction represents a model of freehand computer interaction

VR VR amalgamated both the real and the virtual to create a single sensory experience

# Chapter 1

# Reviewing the field of UGI

Chapter 1.1 introduces the concept of the gesture interface, discussing how human gestures and tools have evolved in parallel. Chapter 1.2 reviews developments in the field of unencumbered gesture interface design. Chapter 1.3 identifies criteria that can be used to determine the potential commercial and ergonomic success of a user interface.

## 1.1 Introduction

First described by Archimedes, the lever and compound pulley are mechanisms that extend the natural capacities of human gesture. Through the use of these mechanisms we are able to multiply the amount of mechanical force that can be applied to physical entities. The lever and compound pulley are some of the earliest examples of encumbered machine interaction. To extend the capabilities of human gestures an array of mechanisms have subsequently been developed. Introduced during the nineteenth century the typewriter became an innovation that made the communication and dissemination of knowledge both clearer and physically sustainable. It has enhanced our ability to use gesture in the production of standard legible print. The analogue joystick is another mechanism that facilitates encumbered machine interaction. Through the use of this mechanism manual hand gestures can be used for controlling machines across the horizontal and vertical axes. This innovation has facilitated the precision control of vehicles

across multiple lines, making the steering of aeroplanes and industrial machinery both possible and efficient. Much of our daily real-world interactions are mediated by gestures, whether they are used to open doors or to grasp a cup to quench a thirst. The evolution of human culture is evidently link to the development and refinement of physical actions. The physical tools that shape these actions underpin technological and cultural development in human society. In essence, the physical tools we utilise in our daily interactions are extensions of human gesture. The advent of the microprocessor has ushered in a new phase of gestural machine interaction. Through its use our gestures are able to manipulate large volumes of information. The microprocessor also enables physical actions to be transmitted over large distances and relayed across the globe. Human actions have been successfully translated into the digital world through the use of peripherals, such as the keyboard, mouse and touchpad. The product and quality of our interactions are affected by their underlying efficiency. Consequently, as interface users we can only operate within the constraints of these devices. The encumbered interfaces described require users to remain proximal to manual devices. These interfaces also limit the freedom of movements of users, such constraints present interface developers with many opportunities to improve human computer-interaction. Increased freedom of movement has been proven to offer significant physiological and performance benefits to users [Zhai et al., 1996]. Through the advancements in computer-vision and pattern recognition it is becoming increasingly possible for human gestures to be optically recognised by digital machines [Bowden et al., 2003; Starner and Pentland, 1995; Zahedi et al., 2005; Zieren and Kraiss, 2005]. Through using this technology people can be liberated from having to be proximal to digital devices. Offering users the chance to control computers through unencumbered means. Optical gesture-recognition has real potential to enable people to engage with computers through fully unrestricted movement of the hand. Interfaces that enable greater hand freedom may also facilitate a greater precision and control of digital devices, such as tools, robots and vehicles [Kortenkamp et al., 1996; Wachs et al., 2008]. Hand-free interaction has the potential to produce more expressive and intuitive interaction. The field of unencumbered gesture-interaction (UGI) is likely to flourish as a result of the increases in computer processing power together with decreases in computer

hardware costs. UGI is also benefitting from advances made in computer programming and encumbered interaction. The challenge facing UGI developers is how best to create sustainable and ergonomic interaction.

### 1.1.1 Mixing virtual and augmented reality

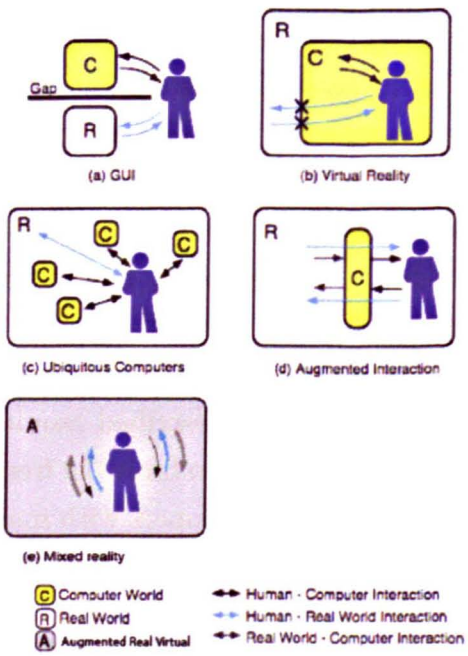



Figure 1.1: (a)(b)(c)(d) Rekimoto and Nagao [1995] illustrate real and virtual interaction. (e) Leon Barker illustrates mixed reality

The desire to create multisensory machine interaction has been a long held aspiration for developers and advocates of Virtual-Reality (VR) and Human-Computer Interaction (HCI). Aspirations that can be traced back to the late 1960s when computer interface design was beginning to emerge as a topic of discourse and debate. Early innovators such as Ivan Sutherland and Myron Krueger were beginning to shape the direction of computer interaction. Krueger et al. [1985] believed computers could be used in the school classrooms to train children through the creation of educational simulations. Sutherland [1968] believed that computers could be used to give mathematicians and scientists a tangible

sense of the imperceptible through the use of simulations. Both Sutherland and Krueger defined the concept of virtual-reality (VR) through the creation of a series of prototypes. However, Sutherland's head mounted display and Krueger's VIDEOPLACE define opposite hemispheres within VR. Though both approaches are subtly different the divergence in outcomes are significant. Sutherland's vision of HCI is based on representing virtual worlds and objects to the user. This approach corresponds to a field of VR research called augmented reality (AR). Sutherland and Sproull create an augmented reality head mounted display (HMD), the first AR interface of its kind. Subsequently, a variety of other AR interfaces have been developed, these range from the see-through optical display to the video-coupling display. The HMD is a device that superimposes virtual objects or data on to representations of the real world, presenting the user with a visual synthesis of the virtual and real. For example, the helmet visor worn by military pilots augment mission data with the real world environment. Sutherland's model of VR amalgamated both the real and the virtual to create a single sensory experience. Krueger, a pioneer within the fields of Virtual reality and unencumbered computer-interaction, offers an alternative model of HCI. He introduces the concept of UGI through the creation of the VIDEOPLACE prototype, advocating the use of freehand gesture as opposed to tactile and mechanical user input. Krueger's model of VR represents an archetypal model of Augmented Virtuality (AV), which inhabits the opposite pole to AR in the Reality Virtuality Continuum [Milgram and Kishino, 1994; Tamura and Yamamoto, 1998] . Unlike AR, which focuses on representing virtual spaces and data to interface users, AV represents a practice of augmenting virtual responses to physical input. In the VIDEOPLACE prototype people could use their hands to alter the shape and position of virtual objects. The modern weather forecast simulation is an example of an AV system that uses live satellite sensory data to create a virtual map of global weather patterns. The inherent ambiguity of the Reality Virtuality Continuum has led to the definition of virtual reality being expanded and interpreted as the mixed reality (MR) continuum [Milgram and Kishino, 1994; Tamura and Yamamoto, 1998] . The mixed reality model of VR reflects that there is a range of possible permeations for how Virtual and physical realities can overlap to affect one and other.

Both Sutherland and Krueger utilised contrasting methodologies to achieve their aims. Sutherland's research relies on the use of encumbered technology, where as Krueger advocated the use of unencumbered interaction. To date, encumbered forms of interaction have largely dominated both VR and HCI. To see this dominance, one needs only to examine the wide range of peripheral devices currently utilised within the computing and gaming industries, with interfaces such as the joystick control pad, the mouse and keyboard. Computer interaction has not deviated from the model of interaction developed by Sutherland [1964]. Four decades after the introduction of VR and UGI, personal computing has continued to develop along the model of graphic user interaction (GUI) via a windows-icons-menu-pointing (WIMP) interface, with a keyboard and mouse. Such interaction provides basic mechanical augmented reality at best. Recent advancements in signal-processing algorithms coupled with the reduction of in microprocessor costs increase the economic viability for novel forms of augmented-virtuality and augmented-reality interaction to be developed. A chronological review of developments of unencumbered AR interaction from its inception in 1970 to the present is documented in the following section.

## 1.2 Critical review of unencumbered gesture-interfaces

This review will discuss developments made within the field of unencumbered gesture interaction. Looking primarily at the development of modes of interaction that facilitate hands free interaction; where the user is not encumbered or physically burdened with tactile or haptic control mechanisms, such as in Keyboard, joystick or mouse interaction. The aim of this review is not simply to consider the merits of novel user-interfaces; it is intended to investigate the types of apparatus that might effectively facilitate the implementation of a robust and ergonomic mode of unencumbered human-computer interaction.

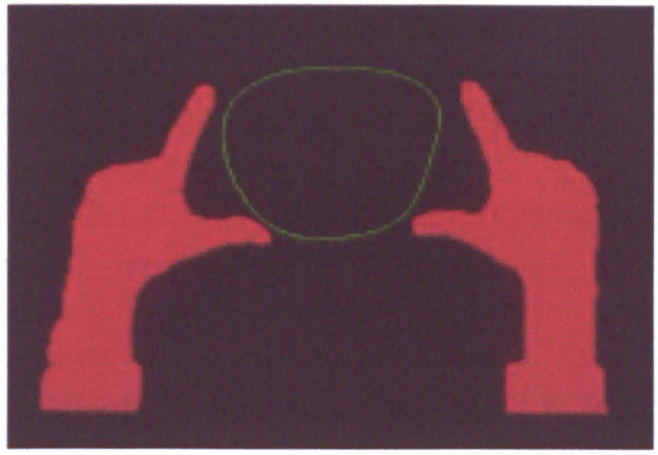


Figure 1.2: Krueger et al. [1985] demonstrates a means of modifying a B-spline curve with the hand

#### 1.2.0.1 VIDEOPLACE

As previously discussed (1.1.1) the VIDEOPLACE developed by Krueger in 1970 is a pioneering form of UGI. In the VIDEOPLACE interface the user's silhouette is captured against a plain background and digitised. The digitised silhouette is then superimposed into a mixed reality environment. Through using their physical actions the user is then able to affect virtual objects within the scene. This interface presents one of the earliest forms of computer-vision mediated interaction. Krueger expands the VIDEOPLACE concept with the creation of the VIDEODESK. The VIDEODESK is a ceiling mounted camera that monitors the user's hand as they rest on a desktop. The user can then utilise their hand gestures to engage with a range of applications. For example, the user can use their hands to make and reshape objects in order to create virtual sculptures. By creating these interfaces Krueger demonstrates how deictic, ergotic and iconic gestures can be used for manipulating virtual objects. Using these methods Krueger illustrates that the thumb and forefinger can effectively be used to modify a B-spline curve through using a limited number of control points (Figure 1.2 page 6). In this way, the VIDEODESK system facilitates object and graphic modelling via gesture. The VIDEOPLACE is a pioneering, but simple, real-time application that demonstrates the use of a straightforward gesture command structure. This

model of computer interaction mediated by clearly defined gesture syntax may become the archetypal method of facilitating unencumbered gesture interaction. As discussed (chapter 2 pages 36 - 38) , the development of such commands has become a significant area of debate with conflicting ideals, with some researchers advocating the use of natural and instinctive gestures. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 20).

#### 1.2.0.2 'Put that there' and 'Put that where'

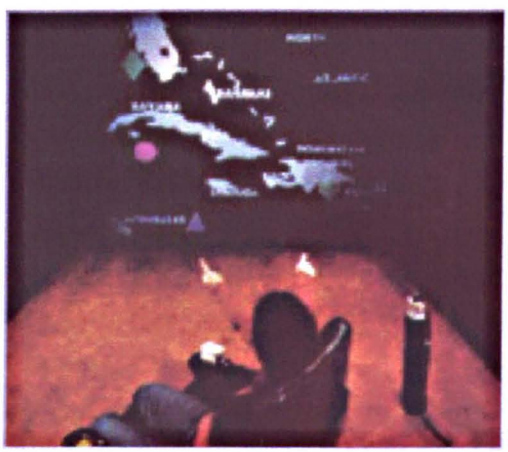


Figure 1.3: Bolt [1980] 'Put that there' interface

The 'put that there' interface, developed by Bolt [1980] , is one of the first multimodal interfaces to combined both speech and pointing gestures in computer interaction. To convey the deictic pointing gesture users of Bolt's interface wore a data-glove. This mode of interaction is encumbered in nature, however Billinghurst and Kato [2002] later created an unencumbered version that utilises computer vision. This later model is called 'put that where'. In the creation of these interfaces both Bolt and Billinghurst presents a strong case for combining the modalities of gesture and speech. Both demonstrate that gesture and speech can be combined to create a powerful interface. The voice can interact with virtual objects regardless of whether they are hidden or occluded from view in an on-screen environment and gesture allows us to intuitively investigate and

manipulate objects in a direct way, each modality complements the deficiencies of the other. There is sufficient evidence that supports the conclusion that users overwhelmingly prefer a combination of voice and gestural interaction. Seventy one percent of users tested in a survey, conducted by Hauptmann and McAvinney [1993], suggested that multimodal speech and gesture input was preferable to using independent modalities. Despite this evidence such interaction is not yet commonplace even though speech recognition has been implemented into a range of commercial operating systems. The combination of speech and encumbered gesture has found usage with military applications such as in the Euro-fighter Typhoon flight cockpit, where maintaining uninterrupted sight of other vehicles and geographical features can be critical. The use of speech recognition is a feature aimed at reducing a pilot's physical workload. Developing multimodal interfaces will enhance our interactions with computers. However, as discussed in chapter 2 (page 39) there are significant concerns regarding the physical sustainability of speech recognition interfaces. Furthermore, there are situations where the use of speech is not practical or desirable. Using speech interfaces within a busy environment might create very loud working conditions. As a result such an interface would be inappropriate for use in some places, such as libraries. A speech interface would also be inappropriate for accessing and documenting personal and confidential information, especially in public places. Speech interfaces would also be inappropriate to use in noisy environments, as ambient noise can diminish speech recognition accuracy. The inherent issues surrounding the use of speech interfaces suggests a framework that enables each modality to operate independently remains essential to providing interfaces that cater to a broad range of human affordances and preferences. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 22).

#### 1.2.0.3 'The DigitalDesk calculator '

The DigitalDesk calculator is an interface that responds to multi-sensory input. Combining pointing gestures with text recognition the DigitalDesk integrates a digital virtual desktop with a real-world desktop. This type of interface conforms

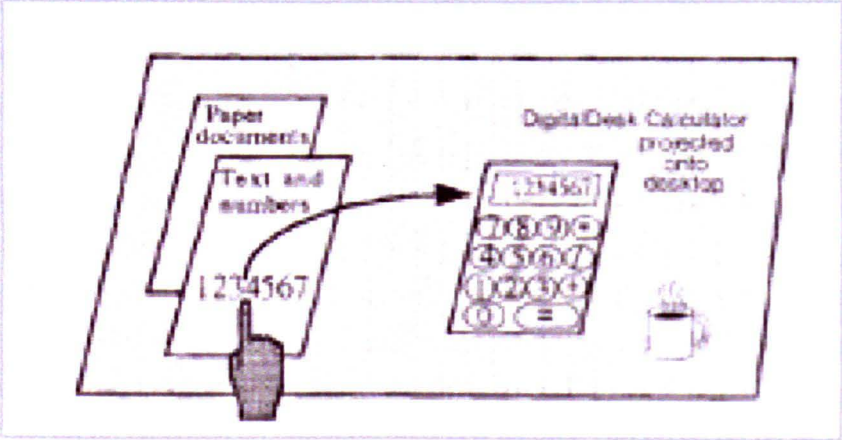



Figure 1.4: Wellner [1991] DigitalDesk calculator

to Milgram and Kishino [1994] description of both the augmented real and virtual. The interface utilises a ceiling mounted video camera and data projector, which have been calibrated to focus on the surface of a table. When activated, the virtual desktop environment is projected on to the physical desktop to create a hybrid, mixed reality. Any document placed upon the desk becomes an interactive element in the newly created space. Direct interaction takes place using deictic pointing gestures. For example, scrolling down the document with the finger will highlight passages of interest contained within the text. Through the use of text recognition the selected text can then be transcribed into digital form. Wellner [1991] has essentially managed to integrate an optical scanner into the desktop environment and creates an efficient and intuitive means of selecting and manipulating text. Although the DigitalDesk receives multi-sensory input, the full potential of gesture is not utilised. Wellner's study identified issues with determining whether a gesture is actively selecting text or simply moving around the desktop. These problems were a result of recognition problems, which could have been solved through the use of better detection algorithms or the use of more distinctive gestures. The problem of recognition ambiguity encountered by Wellner highlight the potential need for a gesture efficiency dataset that documents the recognition accuracy of gestures. Such a dataset would enable developers like Wellner to create interfaces with recognisable vocabulary syntax. Other issues

that affected the efficiency of Wellner's interface involved the physical layout of the Digital desk interface. The users physical position placed them between the projector and the table surface, as a result users were wary of their shadow effecting and occluding the GUI. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 23).

#### 1.2.0.4 Television control by hand gestures

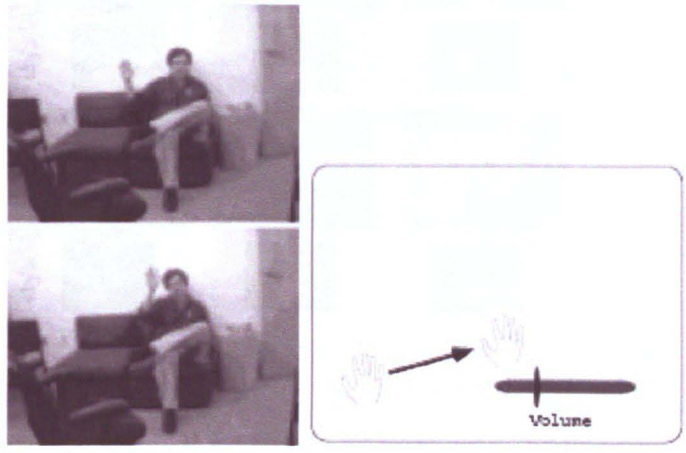



Figure 1.5: Freeman et al. [1995] TV hand control interface

In an attempt to replace the television remote control Freeman et al. [1995] developed an unencumbered gesture interface that enabled viewers to adjust and control their television sets. Using a mixture of symbolic and deictic gestures this interface enables similar interaction to that of a computer mouse. To initialise control of the television the viewer holds up his or her hand so that it faces the screen and performs a trigger gesture. After performing this gesture a graphic sliding control is superimposed on the bottom of the screen. Using the same hand posture the viewer can then control the slider, by moving their hand from left to right. The viewer is able to monitor a graphic representation that corresponds to the motion of their hand and a simple form of augmented visual feedback is presented to the viewer. In demonstrating this interface Freeman illustrated the potential for unencumbered gesture interaction to be employed in real world ap-

plications. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 23).

#### 1.2.0.5 A consumer electronics control system

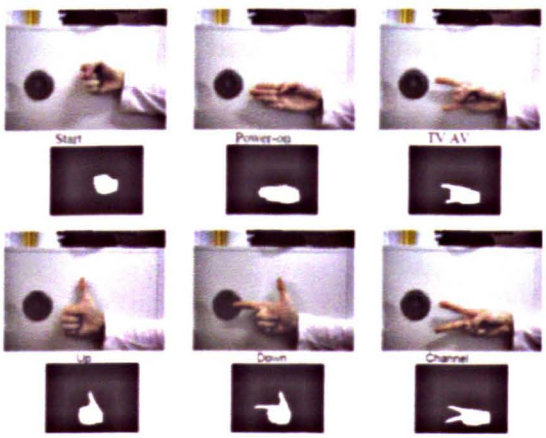



Figure 1.6: Premaratne and Nguyen [2007] gesture set

Premaratne and Nguyen [2007] produced a more efficient and robust version of Freeman et al. [1995] Television hand control interface. In this interface hand postures are captured using skin colour segmentation. This is a method that enables the hand to be recognised against coloured backgrounds. Premaratne and Nguyen identified seven gestures that are distinctive and easily recognisable by the system and designed a simple gesture syntax that can facilitate interaction with consumer devices such as video recorders and television sets. The finite number of distinctive gestures in Premaratne and Nguyen's gesture set enabled it to be embedded into electronic devices that can be positioned adjacent to home appliances. Their system has the potential to be one of the first unencumbered devices to go into mass production. The syntax created for this interface contains gestures that can be evaluated through examining the GEf dataset compiled in Chapter 4 and documented in appendix 1.1 and 1.2. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 24).

#### 1.2.0.6 A gesture operated mobile robot

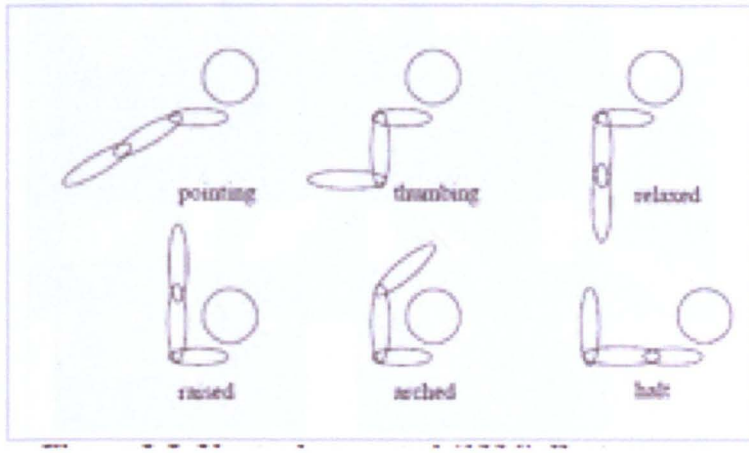



Figure 1.7: Kortenkamp et al. [1996] Gesture command

Kortenkamp et al. [1996] developed a multimodal interface for controlling a mobile robot with a combination of speech, deictic and symbolic gestures. Developed to work alongside humans in challenging environments such as space, underwater or on the battlefield where ambient noises often reduce the quality of speech interaction. The robot is equipped with the ability to clarify speech commands by recognising accompanying gestures. Utilising gestures to support voice commands the mobile robot interface can be controlled within noisy environments. In the development of this interface Kortenkamp et al identified that geometric information can be communicated with greater ease through the use of gestures. A gestural instruction set containing six distinct gestures was developed, maximising natural forms of human communication in a remote control interface. The gesture operated mobile robot (GOMR) created by Kortenkamp et al demonstrates further the potential of unencumbered gesture interaction. Unencumbered interaction has the potential to facilitate the control of vehicles and robots, through the use of a clearly defined gesture vocabulary and syntax. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 25).

#### 1.2.0.7 The Graylevel VisualGlove

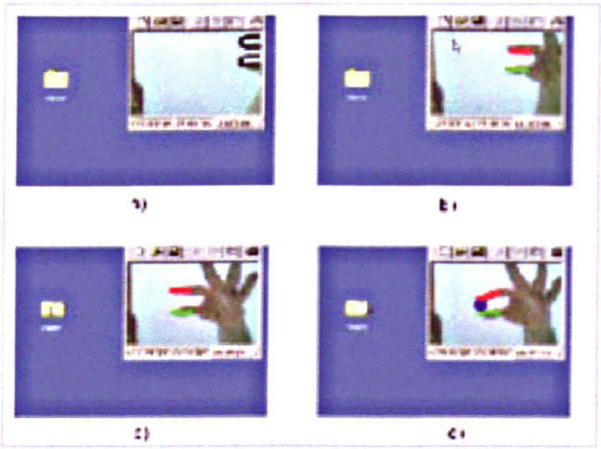


Figure 1.8: Iannizzotto et al. [2001] Graylevel VisualGlove

The Graylevel VisualGlove developed by Iannizzotto et al. [2001] is another example of unencumbered gesture interaction mediated by computer vision. Due to the increase in processing power the modern CPU offers computers are becoming smaller and more portable. The reduction in size of the computer interface poses significant questions to interface designers, as to what types of interface should be used for interacting with small devices. Iannizzotto identifies this issue in the development of his VisualGlove interface. He suggests that by integrating an optical computer display into a pair of spectacles, gesture can be used for interacting with small devices. By using deictic and ergotic gestures to create a point and click interface, an intuitive freehand mouse was created. The modified spectacles are not a necessary requirement and the VisualGlove can be utilised within the context of the traditional screen display. Despite the inherent limitations of point and click interaction, Iannizzotto presents a practical solution to gesture based GUI interaction. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 25).

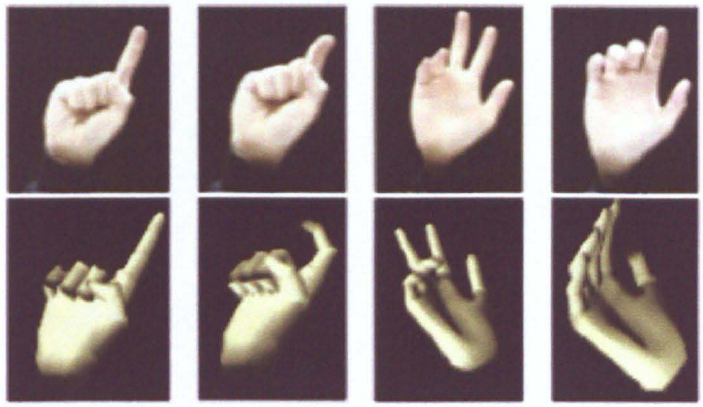


Figure 1.9: Lin et al. [2000] modelling the constraints of the human hand

#### 1.2.0.8 Hand Shape Estimation

Confronted with the problem of programming a computer to visually recognise multiple hand postures, Lin et al. [2000] recognise that the physiological constraints of gesture is a significant aspect to understand if a accurate method for recognising gesture is to be developed. In his research he develops predictive models, which use information about the position of certain fingers to estimate the shape of the whole hand. Lin demonstrates that an accurate 3D model can be produced from 2D images of the hand. Though this method takes into account the physiological constraints of hand gestures it primarily serves to enable greater gesture recognition accuracy. Through the use of this method hand shape can effectively be approximated.The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 26).

#### 1.2.0.9 Light Widgets

Developed by Fails and Jr. [2002], the Light Widget operates by merging a virtual digital environment with physical analogue spaces. The Light Widget interface is an example of an MR interface, as is mixes a physical analogue space with the virtual digital environment. Though this interface shares similarities with the Digital Desk calculator there are significant distinctions between each interface.

Figure 1.10: Fails and Jr. [2002] Light Widgets

Whilst the DigitalDesk calculator visually superimposes the virtual environment into the real environment using a light projection, Fails' Light Widgets manages to superimpose the control functions of electronic devices on to real objects. Fails demonstrates, for example, how the volume control of a radio can be superimposed on to a bed post, enabling the user to alter the volume of the radio by simply moving their hand up or down the post. Using computer vision, the light widget interface contextually relates human physical interaction with real-world surfaces with the operation of electronic devices. In the creation of this interface Fails presents a model of fully immersive interface that utilises natural gesture. The Light Widget is a post-desktop model of HCI and represents an interface with the potential to change our relationship to electronic devices. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 26).

#### 1.2.0.10 Visual Interpretation of sign language

Bowden et al. [2003] develops a range of techniques for optically recognising manual sign language. In developing this framework he highlights the advantages of working with established syntax. Using Stokoe [1960] notation model HA, TAB, SIG and DEZ, Bowden develops algorithms for optically recognising British sign language (BSL). Bowden bypasses issues regarding the development of ergonomic

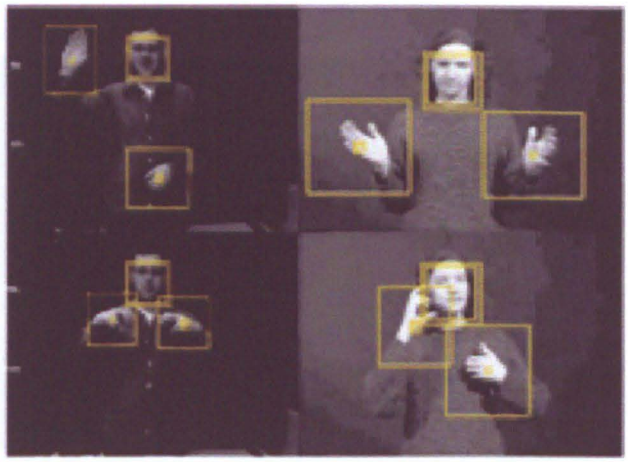


Figure 1.11: Bowden et al. [2003] BSL recognition system

syntax, by focussing specifically on developing detection algorithms. The development of ergonomic syntax specific to computer interaction is however a crucial area of research (see chapter 2 page 35). Bowden employs training algorithms to create probability models from statistical image data. He uses the models generated as templates for detecting the presence of a gesture in an image. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 27).

#### 1.2.0.11 Multi-touch interface

The multi-touch interface is an innovation that has significant potential to revolutionise computer interaction. Developed by Han [2005*b*], the multi-touch interface offers a complete alternative to the current desktop workspace paradigm. Under the system the user is able to directly interact with a GUI, however unlike the DigitalDesk, the user does not have to worry about the effect of their shadow. Hans eliminates the need for peripherals, like the mouse or the keyboard, integrating user input with screen output. The current system uses a mixture of symbolic and deictic gestures. Early incarnations of this interface utilised frustrated total internal reflection (FTIR), a phenomenon that occurs when light travels through a medium such as glass. Light transmitted across the length of a pane of glass internally reflects within the pane at regular wavelengths. The contact of an

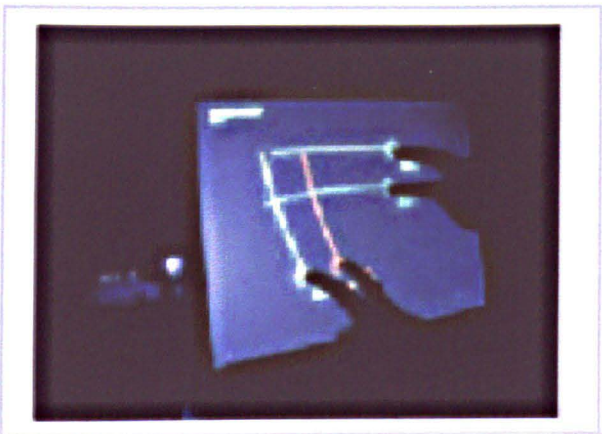


Figure 1.12: Han [2005*b*] Multi touch interface using FTIR techniques

external body, such as a hand or finger, disrupts this and causes light to be released from the glass in the region where contact is made. Hans exploits this phenomenon by channelling infrared light through glass and monitoring the escaping light with an infrared camera. Despite using computer-vision to recognise gesture this interface currently represents a form of semi-unencumbered interaction SUI, as physical contact with a screen interface is required. However, the multi-touch interface shows great promise in being the interface that ushers in the next paradigm of computer interaction. Interfaces that utilise the principles developed by Hans have subsequently entered the marketplace, the Apple iPhone (Apple Inc. 2006), and the Microsoft Surface Interface (Microsoft 2007) represent two such examples. The use of the multi-touch interface has enable both Apple' and Microsoft ' to abandon the need to incorporate a physical keyboard into there respective multi-touch interfaces. In respect to Apple's iPhone this has enable them to increase to size of the visual display screen and incorporate a dynamic customisable software keyboard into their interface. The dynamic nature of the keyboard created enable user's to customise the keyboard to their own specifications and thus improves the ergonomics of their interface. The customisable interface is very attractive to interface developers and it is increasingly likely that mobile devices will adopt similar models of interaction. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter

(see page 27).

#### 1.2.0.12 Thumb and forefinger interface (TAFFI)

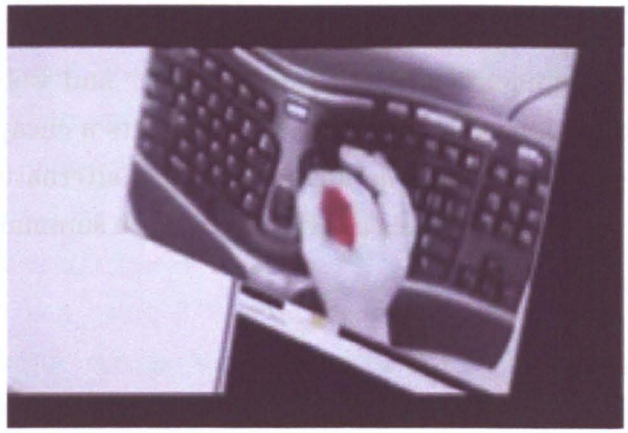


Figure 1.13: Wilson [2006] Thumb and forefinger interface (TAFFI)

The thumb and forefinger interface (TAFFI) is an interaction system developed at Microsoft Podtech research laboratory by Wilson [2006]. The TAFFI uses a pinch detection technique as the basis for unencumbered gesture input. Utilising a set of symbolic gestures, the TAFFI maps the movement of the pinched hand to an on-screen cursor. Input is received via a web camera, which is positioned to capture the topology of the computer keyboard. The image received is analysed and stored. Placing a hand between the camera and keyboard and pinching creates a new shape within the image. The centroid of the new shape can then be tracked and interaction can be created. The pinch interface is similar to the multi-touch interface developed by Han [2005*b*], in that it enables multiple point interaction. However, Han's interface requires physical tactile interaction, whereas Wilson's interface works independent of touch. Wilson's interface adopts an innovative approach to the problems associated with gesture recognition, instead of focussing upon hand recognition Wilson has developed an interface that detects shapes. However, there are limits to the amount of pinch shapes that can comfortably be made with the hand. An interface that relies upon pinch shape detection alone can only facilitate a limited range of interactions. A significant

limitation of the current TAFFI prototype is its reliance upon a static background image, which makes it inappropriate for use with small mobile devices. Devices, such as the iPhone' are increasingly commonplace and are very rarely static when in use. Han [2005*b*] interface allows direct interaction with the GUI, making it suitable for mobile devices. The main advantage the pinch interface offers is that no specialised equipment, other than a computer and web camera, is required. Despite the limitations described, the TAFFI offers a cheap and practical model of unencumbered gesture interaction and a viable alternative to the mouse. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 28).

#### 1.2.0.13 Gestix

Figure 1.14: Wachs et al. [2007] Gestix interface

Developed by Wachs et al. [2007], Gestix is an interface developed for use in a surgical environment. Using symbolic gesture, this interface is designed to give surgeons hands-free access to the computers and visual displays used in an operating theatre. Such a model of interaction would help surgeons and doctors

reduce the potential for cross contaminates and infections to be spread through contact with medical equipment. Using gestures to navigate pie menus and replicate symbolic instruction, the Gestix interface allows surgeons to comfortably navigate a WIMP interface. Wachs et al correctly identify the need to evaluate a gesture's efficiency. They do not, however, explicitly identify the need to measure how uniformly multiple users utilise particular gestures when executing specific tasks. Furthermore, the measures Wachs et al. [2007] define are weighted in favour of the proficiencies of computer vision rather than physical preferences of computer users. Like most UGI research, Wachs et al derive their findings from the analysis of 2D image datasets. The potential for visual occlusion of fingers to occur when using 2D image representations limits the overall accuracy of optical gesture recognition. Using this approach semiotic postures, such as the thumbs-up-sign, have been extensively utilised as they represent postures that can most accurately be recognised through 2D image processing. Though Wachs et al present a framework for measuring a gesture's performance, they admit limitations in their method for evaluating the intuitiveness and comfort of a gesture. The strengths and weaknesses of this interface have been summarised in the conclusion of this chapter (see page 28).

### 1.2.1 Strengths and Weaknesses

As mentioned at the beginning of the section (page 5) this review is primarily concerned with examining the feasibility of creating ergonomic unencumbered models of computer interaction. The following section will outline the legacy of the interfaces examined during this review, specifically highlighting the potential benefits these might offer future computer users.

#### 1.2.1.1 VIDEOPLACE (page 6)

Innovations:

- A pioneering form of UGI that enables users to interact with projected graphics using their gestures

Strengths:

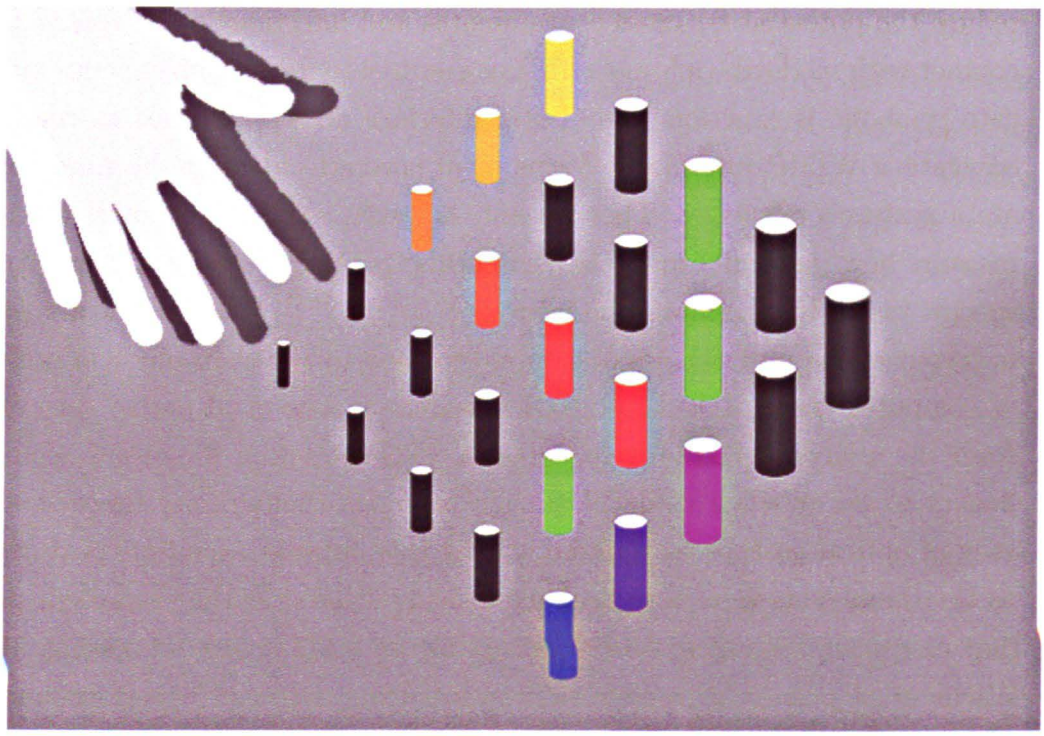


Figure 1.15: Illustration of a silhouette occluding a graphic interface

- An innovation that has the potential to facilitate the creation of dynamic updateable(sic) graphic interfaces.
- The interface enables direct manipulation of graphic models through the use of the hands, employing straight forward gesture commands.
- This early prototype has great potential for the technology to be improved through novel image processing and computer-vision algorithms.

Weaknesses:

- The configuration of interface components allowed the interface user's actions to occlude the graphic display (see figure 1.15), meaning the users presence negatively affected their own view of the graphic interface
- The interaction between the user and interface could also be improved by the use of emerging techniques such as segmentation using motion and depth mapping (see chapter 3 pages 63 - 66).

- The physical construction of the interface limits the overall portability of the system
- Such an interface configuration might be acceptable for general purpose non-critical applications. However, for systems requiring precise interaction where errors maybe costly the occlusion of the graphic interface maybe unacceptable.

#### 1.2.1.2 "Put that there" and "Put that where" (page 7)

Innovations:

- A multimodal interface that facilitates users with the ability to utilise both speech and gesture in a single interface.

Strengths:

- An innovative multimodal form of interaction that enables users to both utilise speech and gesture when interacting with a computer.
- The interface enables direct interaction with drop-down menu interfaces through the use of speech and pointing commands.
- The interface enables users to interact with visually occluded regions of the on-screen environment via the use of speech input.
- There is evidence to suggest that multimodal speech and gesture input is preferred by computer users as opposed to the use of independent modalities.

Weaknesses:

- Potential to negatively impact users as a consequence of voice loading (chapter 2 page 39).
- The speech component of this multimodal interface may become less effective in noisy public environments, such as libraries, cafe and outdoor urban spaces.

#### 1.2.1.3 "The DigitalDesk calculator " (page 9)

Innovations:

- Successfully creates a hybrid real-virtual working environment that integrates the real world office desk with the virtual digital workspace environment.

Strengths:

- The interface enables users to treat passages of printed text on paper as you would text stored in digital memory, eliminating the need to hand type un-digitised text.
- Translates real world information into digital data.

Weaknesses:

- The digital desk suffers from a limited set of gestures
- The configuration of the interface meant that the users presence negatively affected their own view of the graphic interface
- The physical construction of the interface limits the overall portability of the system

#### 1.2.1.4 Television control by hand gestures (page 10)

Innovations:

- Utilising a mixture of symbolic and deictic gestures to create a simple interface for controlling the volume of a television set this application successfully demon- strates the feasibility of utilising computer-vision and gesture recognition in a domestic setting.

Strengths:

- The suggested interface offers users the opportunity to replace the standard television remote control with their own gestural actions.

- Such an interface would eliminate the need to buy replacement batteries for the remote control and prevent people from having to locate a sometime elusive control device.
- Successfully demonstrates the feasibility of using gesture recognition in a domestic setting using consumer devices.

Weaknesses:

- No mechanism for determining the primary user.
- The interface only has a limited set of gesture commands

#### 1.2.1.5 A consumer electronics control system (page 11)

Innovations:

- This interface facilitates users with the ability to utilise seven symbolic gestures to interact with consumer devices such as video recorders and television sets.

Strengths:

- Similar to Freeman et al. [1995] TV hand control interface this interface offers users the opportunity to use their own gestural actions to command remote controlled consumer devices.
- Offers improved gesture recognition to that of Freeman et al. [1995] TV hand control interface.
- Successfully demonstrates the feasibility of using gesture recognition for operating consumer devices.
- The gesture set created was optimised sufficiently to allow the detection algorithm to be embedded into electronic devices that can be positioned adjacent to home appliances..

Weaknesses:

- No mechanism for determining the primary user.
- The interface only utilises symbolic gestures.

#### 1.2.1.6 A gesture operated mobile robot (page 12)

Innovations:

- Provides a multimodal method for interacting with robots and machines in challenging environments.

Strengths:

- An system capable of being utilised in challenging environments such as space, underwater, heavy industry or on the battlefield.
- Provides a mechanism for controlling robots through a combination of speech and gestures
- Demonstrates the feasibility of using gesture recognition and computer-vision for controlling precision critical devices.
- Illustrates that systems can be developed to accurately distinguish between a range of gestures.

Weaknesses:

- The potential to negatively impact users as a result of voice loading is a significant factor limiting the overall ergonomics of this communication interface (see chapter 2 page 39).

#### 1.2.1.7 The Graylevel VisualGlove (page 13)

Innovation:

- This interface provide a method for interacting with small device using unencumbered gesture interaction.

Strengths:

- Introduces a novel approach for interacting with small mobile devices, which could potentially allow the facades of such devices to be dedicated to displaying graphic output.

- Facilitate drag-and-drop functionality without the use of a computer mouse

Weaknesses:

- The practicality of using unencumbered gestures to control a handheld mobile device are yet unproven. Such a device is unlikely to out-perform a mouse or touch based interface.

#### 1.2.1.8 Hand Shape Estimation (page 14)

Innovation:

- Recognises that there are physiological constraints within human hand anatomy that prevent fingers from moving independently, an approach for estimating hand posture was developed.

Strengths:

- Useful method for generating 3D models of hand posture through using 2D images
- Could be expanded to facilitate full body shape estimation.

Weaknesses:

- Though this could be a useful approach to utilise in a gesture recognition system it is not a complete interface.

#### 1.2.1.9 Light Widgets (page 15)

Innovation:

- Using computer vision, the light widget interface contextually relates human physical interaction with real-world surfaces with the operation of electronic devices. This interface demonstrates how the volume control of a radio can be superimposed on to a bed post, enabling the user to alter the volume of the radio by simply moving their hand up or down the post.

Strengths:

- Creates a fully immersive interface that utilises natural gesture.
- Present post-desktop model of HCI.
- Presents an interface with the potential to change our relationship to electronic devices.

Weaknesses:

- Needs to be in a fixed location so is not portable.

#### 1.2.1.10 Visual Interpretation of sign language (page 16)

Innovation:

- Develops a range of techniques for optically recognising a large lexicon of manual sign language.

Strengths:

- Publishes a wide array of approaches and techniques that can be utilised by developers to create computer-vision systems capable of recognising symbolic gestures.

Weaknesses:

- Lexicon only includes symbolic gestures

#### 1.2.1.11 Multi-touch interface (page 17)

Innovation:

- The multi-touch interface offers a complete alternative to the current desktop workspace paradigm. Under this system the user is able to directly interact with a GUI using touch interaction.

Strengths:

- User is able to directly interact with a GUI.
- System uses a mixture of symbolic and deictic gestures.

- The dynamic nature of the keyboard created enable users to customise the keyboard to their own specifications and thus improves the ergonomics of their interface.
- Introduces a novel approach for interacting with small mobile devices, which could potentially allow the facades of such devices to be dedicated to displaying graphic output.

Weaknesses:

- Does not facilitate a fully unencumbered mode of interaction

#### 1.2.1.12 Thumb and forefinger interface (TAFFI) (page 18)

Innovation:

- This interface maps the movement of the pinched hand to an on-screen cursor, via the use of a web camera.

Strengths:

- Does not require specialised hardware
- Offers a cheap and practical model of unencumbered gesture interaction and a viable alternative to the mouse

Weaknesses:

- An interface that relies upon pinch shape detection alone can only facilitate a limited range of interactions.
- The prototype relied upon having a static background image, which makes it inappropriate for use with small mobile devices.

#### 1.2.1.13 Gestix (page 19)

Innovation:

- Using symbolic gesture, this interface is designed to give surgeons hands-free access to the computers and visual displays used in an operating theatre

Strengths:

- Such a model of interaction would help surgeons and doctors reduce the potential for cross contaminates and infections to be spread through contact with medical equipment.
- Using gestures to navigate pie menus and repli- cate symbolic instruction, the Gestix interface allows surgeons to comfortably navigate a WIMP interface.
- Develop an optimised gesture vocabulary

Weaknesses:

- Potential issues regarding the ergonomics of the set of gesture commands used in the interface.

## 1.3 Summary of review findings

This review illustrates that varied and dynamic modes of interaction can be facilitated through the use of computer-vision and optical gesture recognition. The varied range of prototypes demonstrate that there is great potential to facilitate unencumbered human computer interaction through the use of computer-vision. The VIDEOPLACE developed by Krueger is a definitive example of how gestures could be used in unencumbered computer interaction. Krueger demonstrates that a broader range of actions than mechanical button input can be recognised by computers and utilised in the operation of a computer application. These examples illustrate that there are a multitude of potential applications that would benefit from the implementation of robust ergonomic UGI systems. Though the prototypes discussed earlier in this chapter demonstrate successful proof of concept, they also highlight areas that need development. This review has identified three important criteria, which can be used to determine the success and viability of a gesture interface

### 1.3.1 Three steps to the development of a viable gesture interface

The first important element of an interface is the establishment of a coherent platform, with a cohesive framework that limits potential for negative feedback to be experienced by the user. For example, an interface that is uncomfortable and cognitively challenging represents a failure of this criterion. The second requirement for success relies on whether an interface fulfils concrete needs or functions. The need might be completely novel, as with the Gestix interface. Alternatively it may simply be an improvement on a previous interface. Either way, the user will need to believe that they will benefit from the adoption of an interface. The third criterion refers to the gesture syntax required to operate the interface sustainably and efficiently. For example, the procedure required to effectively operate and complete a task has to be clear and repeatable. An example of an interface that succeeds in fulfilling all three criteria is Han [2005*a*] multi-touch interface. The success of this semi-unencumbered gesture interface can be attributed to the design of the physical interface and that it allows users greater amount of customisation than other handheld devices. The design and customisable nature of its physical interface means that it fulfils the first criterion identified. This interface offers a significant improvement to previous GUI's on small devices, which as a result of previously having a fixed inflexible keyboard compromised by reducing the overall size of the visual display. This improvement upon previous interfaces means that the second criterion is also fulfilled. In addition to these criteria the Han [2005*a*] interface benefitted from integrating the syntax previously developed by Westerman [1999], a syntax developed specifically for the purpose of being intuitive and ergonomic. Thus the third criterion is also met. In contrast, the DigitialDesk developed by Wellner [1991] though it represents a triumph in the advancement of UGI research it failed to fulfil any of these criterion. One reason for this failure is the lack of cohesion between the various elements of the interface. For example, the user of DigitialDesk interface was sandwiched between the projected display and the GUI, as a result the user casted a visual shadow over the interface. The composition of this interface creates a negative feedback out of the user's physical presence. As a result this interface fails to

fulfil the first criterion. The DigitalDesk failed to fill a concrete need or improve a previous mode of interaction and thus does not meet the second criterion either. The DigitalDesk calculator also suffered from having ambiguous gesture syntax, determining whether a gesture was actively selecting text or simply moving around the desktop were problematic. Consequently this interface fails to fulfil the third criterion also. With the right interface technology and syntax the DigitalDesk might find greater success in commercial HCI applications. However, without addressing these stated criteria the concept of the DigitalDesk calculator as presented by Wellner will not find widespread usage. The Gestix interface produced by Wachs et al utilises a viable interface technology and is able to fulfil the first criterion discussed. In addition, the Gestix interface is an example of a fully unencumbered gesture interface that has the potential to play a significant role in medical robot control, thus it fulfils the second criterion. However, though the vocabulary and syntax they utilise is accurately recognised by computers it is not user centric and as a consequence may prove physically unsustainable. The Gestix interface demonstrates that fully unencumbered interfaces can be utilised in precision critical modes of interaction.

The most significant obstacle preventing UGI platforms from being successfully utilised in commercial HCI applications is the failure to fulfil the third criterion. The establishment of an ergonomic gesture syntax that can be accurately recognised by computers and sustainably performed by users is essential to the development of UGI. Though there is a significant amount of research into the physiology of gesture there is little understanding regarding the sustainability of UGI, as most UGI research prioritises recognition accuracy over the physical preferences of users [Nielsen et al., 2004; Pavlovic et al., 1997] . Grounding gesture syntax development around the limitation of current technologies will only extend the cycle of obsolescence to incorporate UGI syntax in addition to computer hardware. It is understandable that interface developers shape human computer interaction around the capabilities of computers. However, the development of robust and ergonomic syntax needs to be prioritised and the capabilities and preferences of people placed central to such developments. As advances are made in programming and hardware, the conclusions of techno-centric research will quickly date and be less pertinent. Our physical and cognitive capacities are

Table 1.1: Criteria determining the success of an interface

| | | |
|---|---|---|
| Criteria 1 | Coherent Interface | Limits potential for negative feedback |
| Criteria 2 | Fulfils/Improves function | Novel or improvement on a previous interface |
| Criteria 3 | Sustainable syntax | Efficient method for effectively complete a task |

constant relative to the speed of development in computing. As a consequence, grounding UGI research on the capabilities of people would prove to be the most sustainable approach to interface development. Such an emphasis on syntax and interface development would limit the cycle of obsolescence to apply to UGI hardware, not the underlying gestural syntax. Though the prioritisation of computer recognition accuracy is indicative of UGI research. The future success of the field is dependent on the development of intuitive and ergonomic gestural syntax.

### 1.3.2 The importance of syntax to UGI

As advances are made, the line delineating the augmented real from the virtual are becoming increasingly blurred. Currently, the boundaries delineating the interface from the ambient environment are clearly identifiable. Interactions with analogue devices, such as the keyboard and the mouse, are mediated through the physical contact of a user. For example, devices such as the keyboard or mouse cannot be successfully utilised without physical contact from the user. Consequently, there is little ambiguity regarding whether a user is engaged in interaction with such devices. However, the creation of algorithms capable of recognising human action through unencumbered methods creates the potential for ambiguities to arise. When interaction is no longer mediated by the necessity to touch a physical surface the boundaries separating the user from the interface evaporate. Under such circumstances, the user's body becomes the interface and all of their subsequent actions are interpreted as input. Separating the interface from the user becomes increasingly difficult. The user may still be able to

clearly distinguish virtual objects and AR environments, but the parameters and boundaries of the actual digital AV interface may become visually imperceptible to the user. The resulting AV interface behaves like a second skin, which can either enable users to control digital machines as naturally as they move their own body; or create a potential straight jacket that restricts and inhibits the user's interaction. To reduce user error and minimise the potential for unintended interaction to occur, a clear and unambiguous syntax has to be created. The future of immersive interfaces present significant challenges regarding how to define active and passive interaction. Furthermore, there are conflicting approaches being undertaken when addressing these challenges. Some developers advocate the creation of artificial gesture syntax to mediate interaction with computers. Other researchers believe that developing algorithms capable of completely recognising natural gesture should be the ultimate goal of UGI [Quek et al., 2002; Wexelblat, 1995] (see 2.1 page 38). The latter approach will require either the development of complex and sophisticated algorithms or the use of multimodal speech in combination with gesture interaction. The former approach will demand that the interface user adopt and learn the necessary gesture vocabularies. Current developments in gesture syntax suggest that artificial syntax has a significant role to play in the development of unencumbered interaction. For example, in a short period of time the multi-touch interface, with Westerman [1999] syntax, has gained wide spread usage through the creation of touch screen devices by Apple (Apple Inc. 2006). Though these devices are only semi-unencumbered, with proximal interaction still being a necessary requirement of interaction. These developments demonstrate that unencumbered models of interaction will need to create ergonomic gesture vocabulary and syntax. As the first generation of fully unencumbered interfaces emerge from various computer science laboratories around the world [Bowden et al., 2003; Starner and Pentland, 1995; Zahedi et al., 2005; Zieren and Kraiss, 2005] the need for a more comprehensive gestural syntax becomes more pronounced. The development of sustainable and universal gesture syntax is in the formative stage. In order for syntax to exist beyond the shelf life of current technology, the affordances and preferences of humans need to be primary concerns. The construction of a gesture efficiency database that catalogues the performance of syntax vocabularies will aid the development of robust

and ergonomic gesture syntax. Such developments will accelerate the evolution of dynamic and sustainable forms of computer interaction. To utilise gesture for precision critical systems, such as operating vehicles, robots or surgical apparatus, a dataset of gesture efficiency is essential.

# Chapter 2

# Gesture Research

Chapter 2.1 defines clear taxonomies for describing intrinsic facets of gesture. The psycholinguistics of human gesture is interrogated and diverse impulses that dictate our physical actions are identified. Chapter 2.2 reviews the underlying constraints and biomechanics of human gesture and discusses the cognitive and physical apparatus behind our actions. This section considers what methods are most appropriate to use by unencumbered gesture interface developers, when evaluating physical exertion and user comfort. Chapter 2.3 defines the methodology that will be utilised during this investigation, which evaluates the efficiencies of gestures.

## 2.1 Taxonomy of gesture

Gesture is an intrinsic part of the vocabulary of human communication. Offering similar capacities for expression as speech, gestures also enable people to manipulate and feel their environments. Though our gestures facilitate a broad range of activities there overall capacity to perform physical tasks and actions is limited by physiological constraints. These constraints can prevent certain actions from being comfortable and sustainable. Focussing specifically upon whether ergonomic gesture reliant interfaces can be created, this chapter (pages 41 - 52) examines inherent limitations and prescribes a method for evaluating the performance of UGI interface lexicons and frameworks. In order that gesture can be used in its

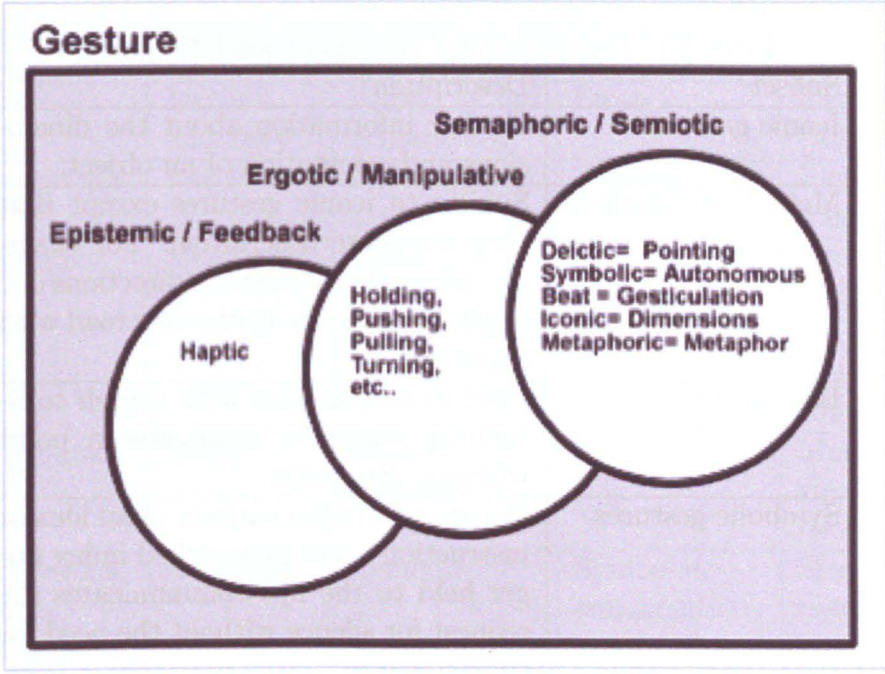



Figure 2.1: Diagram illustrating the various facets of gesture

optimal capacity a detailed understanding of gesture is needed. Psycholinguists such as Mcneill [1992] describe gesture as being a bridge between our conceptualising capacities and our linguistic abilities. This would explain why we often use gesture to communicate what we cannot easily express with words. Researchers like Kendon [1994], Mcneill [1992], Mulder [1996], Efron [1941], Cadoz [1998] Rime and Schiaratura [1991] have devised taxonomies that help foster insight into how we use gesture. Unlike Stokoe [1960] notation model, which describes the appearance of sign language, contemporary research has focussed upon the kind of functions and actions that gestures facilitate. This has led to a better understanding of how and why we use gestures. Cadoz suggests that human gesture can be divided into three major groups the semiotic gesture most often used to communicate information, the ergotic gesture that we use when we manipulate objects in the physical world and the epistemic gestures, which are exploratory and provide us with sensory feedback from our environment. All of these gestures can be broken down into further classifications. Within semiotic these include

Table 2.1: Describing the subsets of semiotic gesture

| Subset | Description |
|---|---|
| Iconic gestures | Convey information about the dimensions and orientation of an object. |
| Metaphoric gestures | Similar to iconic gestures except that they are more descriptive. For example, when giving a person directions one might indicate the course of a road with a gesture. |
| Beat gestures | Used in conjunction with speech to illustrate tempo or emphasise a point within a discussion |
| Symbolic gestures | Communicate the entirety of an idea or instruction - an outstretched index finger held to the lips communicates the request for silence without the need for spoken words. |
| Deictic gesture | Traditionally defined as the pointing gesture, a gesture used to define areas of interest. |

iconic, metaphoric, beat, symbolic and deictic gestures. Within ergotic gesture further categories include holding, pushing pulling and turning. Within epistemic gesture these include haptic feedback that is gained from actions such as touching, feeling and squeezing. (Figure 2.1 page 36). Figure 2.1 illustrates how each gesture can be categorised. Despite having such a broad range of gestures available for study, psycholinguists and UGI researchers tend to focus on the development of semiotic gesture commands. Table 2.1 describes the specific subset of semiotic gestures.

Cadoz. Kendon, McNeil, Rime and Efron present independent taxonomies to describe these facets. However, the taxonomies they identify are similar in most cases. Table 2.2 shows the similarities between these taxonomies. In this thesis the definition of semiotic gesture adheres to the taxonomy applied by Mcneill [1992]. Saluting, using sign language, or thumbing a lift when hitchhiking are all examples of semiotic gestures. The semiotic range of gesture includes: iconic,

Table 2.2: Labels and characteristics of semiotic gesture [Wexelblat, 1998]

| Kendon | McNeill and Levy | Rime and Schiaratura | Efron | Identifying Characteristics |
|---|---|---|---|---|
| Physiographic | Iconic | Physiographic | Kinetographic | Picture the content of speech |
| Ideographic | Metaphoric | Iconic | Ideographic | Portray the speaker's ideas, but not directly the speech content |
| Gesticulation | Beats | Speech-marking | Baton | Marking the rhythm of speech |
| Autonomous gestures | Symbolic | Symbolic | Symbolic/ emblematic | Standardised gestures, complete within themselves, without speech |
| none | Deictic | Deictic | none | Pointing at objects or areas within a space using the hand. |

metaphoric, beat, symbolic and deictic gestures.

Gestures offer a multifaceted way to interact with analogue environments. To fully exploit these facets in computer interaction it would help if a dedicated semantic structure were defined. Human interface developers will increasingly need to examine the findings of gesture research and psycholinguistics, in order that accurate models of interaction can be developed. Research conducted by Quek et al. [2002] and Wexelblat [1998] has made significant progress towards establishing an unambiguous set of taxonomies. Pavlovic et al. [1997] and Wexelblat [1995] suggests that interface developers should focus efforts on implementing natural gesture recognition in computers, utilising gestures already used in everyday encounters, because they are intuitive. However, the gestures commonly applied are used primarily to support and emphasise speech. This suggests that

the use of natural gesture in computer interaction is likely to rely on the use of speech, relegating gesture to fulfil a secondary support role. Though combining the use of gesture and speech in an interface reduces any need for an artificial set of gesture commands, the underlying ergonomics of gesture should still be examined. Although evidence suggests that most users preferred multimodal interaction[Hauptmann and McAvinney, 1993] there is conflicting research that demonstrates some unsustainable aspects of speech interaction. Intensive use of speech recognition systems can stress the speech organs, increasing the risk of conditions such as voice loading being acquired by users [Vilkman, 2000]. Cudd et al. [1998] suggest that such injuries are an inevitable result of using speech recognition technology. The suitability of an interface with such latent potential to damage as critical a function as speech is therefore questionable. Since the hands are currently the primary tools used in HCI, an increase of stress load upon the hands would be comparatively far less than what speech-interaction would place on the vocal apparatus. Furthermore, as previously mention there are situations where the use of speech is not practical or desirable. The use of multiple speech interfaces in busy environments might create very loud working conditions. As a consequence such an interface would be inappropriate for use in some places such as libraries. As a result of these issues, despite arguments put forward by Pavlovic et al. [1997], Wexelblat [1995] and Quek et al. [2002] the development of artificial gesture control syntax that facilitates the control of computers cannot be overlooked . When developing interfaces for gesture interaction it is necessary to understand the physical impact of certain actions. Though it is understandable that interface developers prioritise the performance accuracy of an interface over the longer term cognitive and physical effects on users, this research will prioritise the user. This research will examine factors influencing the viability of unencumbered gesture interaction and examine its capacity to facilitate diverse tasks.

## 2.2 Developing an evaluation framework

The propensities of users to cognitively and physically utilise gesture for unencumbered machine interaction has to be identified if ergonomic models of UGI

are to be created. The following section will discuss some of the cognitive and physical apparatus that facilitate and constrain human gesture. In addition, issues pertaining to the potential benefits of unencumbered machine interaction will also be interrogated. The results of previous usability studies have been assessed in parallel to the development of a methodology for evaluating the performance of gestures. The merits of interfaces such as the keyboard, pen and speech-responsive-interfaces will be discussed and methodologies for developing ergonomic UGI frameworks will also be defined. Addressing these issues will enable the merits of future interfaces to be determined.

### 2.2.1 Gesture cognition

In humans the superior temporal cortices region of the neural system has been demonstrated to show greater activity during spoken language than with sign language. Where as, the posterior middle temporal gyri region shows greater activities during sign language. Despite this both manual and aural languages have been identified as using similar sensorimotor processing apparatus. Comprehension of both signed and spoken languages shows similar activation in both the left superior temporal gyrus and the left inferior frontal gyrus. In addition, both sign and spoken languages have been shown to utilise the mirror neural system, in the left frontal lobe broca region of humans. The mirror neuron system is an important piece of cognitive apparatus that enables people to both observe and execute physical actions [Iacoboni et al., 1999; Rizzolatti and Craighero, 2004; Rizzolatti et al., 2001] . First identified in Macaque monkeys this system is active during the observation of physical action and recreated during the execution of similar actions. The mirror neurons represent a mental model of a physical action, which has either been performed or observed. These sensorimotor processes enable the reproduction and dissemination of complex and subtle physical activities such as speech and gesture. The inherent similarities between the neural networks used in visual, audible and manual languages demonstrate both the complexity and potential of manual communication and interaction.

### 2.2.2 Gesture physiology

When we try to move our fingers individually we are able to observe neighbouring fingers move, to varying degrees, depending on which finger we actively move. If each of our fingers, thumbs and wrists could work independently we would be able to configure each hand into over 4 million subtle arrangements. However, since our fingers share tendons and nerves, individuated finger movement is not possible and the degree of freedom is largely constrained by the interdependence of each finger. In a series of laboratory experiments Schieber [1991]studied interrelated and individuated finger movement. Using rhesus monkeys, Schieber measured the inter-physical relationships of individual fingers when engaged in flexion and extension. The results of this study produced a detailed picture to describe the levels of finger interrelation. The thumb and the wrist were shown to have significantly higher degrees of individuated flexion while the middle, ring and little finger showed the least. The same is true in the case of finger extension, but on the whole all fingers showed less individuation when extended than when flexed. In further studies, Schieber [1995] identified which nerves and tendons were most active during finger motion, creating a map illustrating the interrelation of nerves and tendons in the hand and wrist (See figure 2.2 page 42).

### 2.2.3 Measuring physiological exertion and efficiency

The average person has the ability to type words using a QWERTY keyboard at approximately 35-65 words per minute. This is significantly faster than the 20-30 words per minute handwriting average. The computer keyboard also enables users to easily revise and manipulate text. The combination of these factors means the keyboard offers the reduced risk of repetitive strain injuries (RSI) occurring than when a pen is used. When directly compared with handwriting typing emerges as the more ergonomic word-processing tool. Despite these advantages the risk of RSI associated with keyboard interaction remains, particularly for those who type over 20,000 keystrokes per day [Armstrong et al., 1994; P et al., 1999]. Upper extremity musculoskeletal disorders, such as carpal tunnel syndrome and tendinitis, are some of the conditions that occur as a result of highly repetitive activities like typing [Marklin et al., 1999; Marras and Schoenmarklin, 1993; Moore et al., 1991;

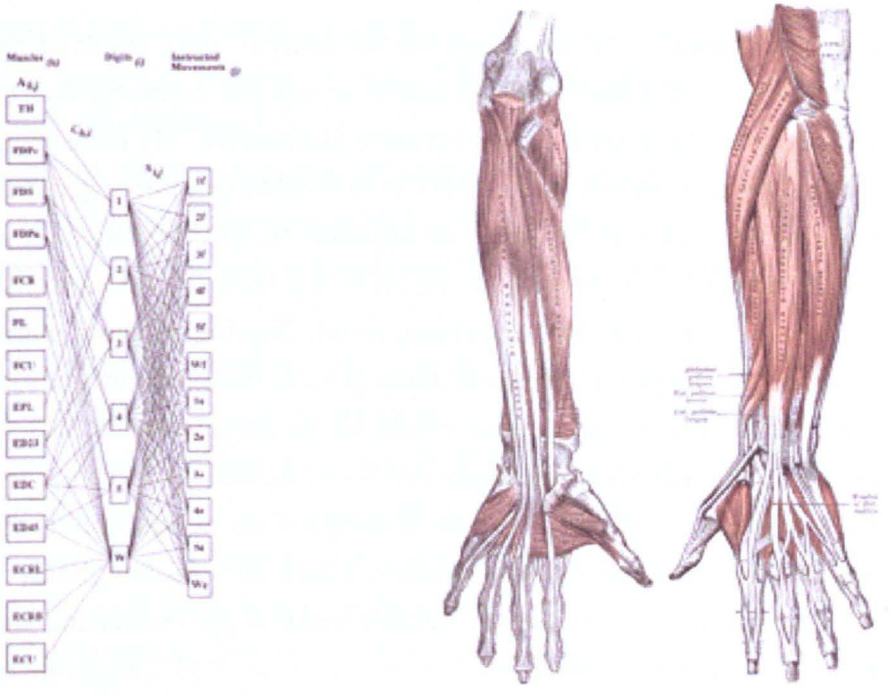


Figure 2.2: Left: Schiebers map illustrating the interrelation of extrinsic finger muscles. Middle and Right: illustration of the nerves and tendons of hands and forearm.

Serina et al., 1999; Sommerich et al., 1996; Wells et al., 1994]. The average fluent signer is able to sign at a rate of 100-120 words per minute placing the performance of an interface based on hand gesture near to that of speech, which affords a rate of 120-170 words per minute. This demonstrates the underlying potential for gesture to enhance computer interaction. However, there remain significant questions regarding the sustainability of gesture-based interaction. There is, at present, limited knowledge of how unencumbered gestural interaction will physically affect computer users. One of the most significant obstacles effecting the identification of sustainable gestures is the difficulty of evaluating muscular activity and physical exertion. Within clinical and biomechanical practice a method often used when assessing physical exertion and muscle activity is electromyography (EMG). This method measures muscle activity by monitoring the small electrical charge that is released by muscles when they are active. Measuring physical exertion in this manner can be very invasive, as this method requires a

needle electrode to be inserted into the muscle. As a result of this process only a finite number of muscles can be tested at any one time. Surface electromyography (SEMG) can be used as a less invasive alternative. By measuring muscle activity from the skins surface the SEMG is only able to offer a general indicator of the muscular activity of the user, as opposed to specific fibres offered by the intramuscular EMG. Subsequently, SEMGs can only provide a limited picture about exertion experienced by the human hand. Neither of these methods represents a practical solution for interface developers or usability evaluators, as a specialist with the necessary equipment would be an indispensable part of the assessment process. Though these methods produces significant information about the muscle activities of users, there are underlying questions about whether particular gestures are effective and sustainable, which are not completely addressed. In order to address these concerns another method for evaluating the performance of a gesture will be investigated. Sommerich et al. [1996] suggests an alternative approach to evaluating physical exertion. Instead of collecting physical data through EMG Sommerich et al. [1996] suggests collecting anecdotal measurements from participants. Using this method Sommerich et al. [1996] demonstrates that an accurate assessment of physical exertion can be obtained. In a study that evaluates the physiological impact of sign language on sign interpreters Scheuerle et al. [2000] employs Sommerichs approach. This study shows that a large percentage of sign language interpreters suffer from cumulative trauma disorders (CTD), such as carpal tunnel syndrome, tendinitis and bursitis. In a survey questionnaire developed to study the degree of pain and discomfort experienced by sign language interpreters, 82 percent of the one hundred and nineteen people surveyed experienced disabling pain or discomfort during and following their occupational tasks. 33 percent of the survey participants experienced pain and discomfort in the hands and wrists. Further studies of sign language interpreters from Quebec document that 81 percent of respondents had experienced shoulder pain during the previous 12 months, 79 percent had experienced neck pain, and 74 percent hand, wrist and forearm pain [Delisle et al., 2005]. These figures contrasted starkly with the 50 percent , 41 percent and 28 percent of pain and discomfort experienced by the general adult population of Quebec, [Daveluy, 2000] this illustrates the acute strain heavy sign language usage can place upon human physiology. This study

also demonstrates that anecdotal information can successfully be used to evaluate the physical impact of gesture on users. The most frequent postures assumed by sign language practitioners [Shealy et al., 1991]include ulnar deviation of the wrist, flexion in the elbows at angles greater than 90 degrees, and pronation of the forearm. The degree of supination and pronation achieved by those engaging with sign language exceeds values found within occupations that represented a high risk of CTD [Marras and Schoenmarklin, 1993]. Together with evidence illustrating the high static load placed upon the back, torso, shoulders and neck we can begin to form a picture as to some of the burdens that sign language places upon the human physiology. Though the investigations discussed in this chapter are insightful they do not provide a direct picture of how gesture can effectively be utilised in computer interaction. Furthermore, these investigations do not provide direct information regarding the potential physiological effects of unencumbered interaction. The lack of research in this area highlights that there is a particular need for a rigorous empirical study into the physical preferences of users. Therefore, a theoretical model of sustainability and a methodology for evaluating the performance efficiencies of gestures will have to be established. To limit the potential for unsustainable paths of development to occur in widespread interaction, an optimal model of UGI needs to be clearly defined. It is therefore important that a methodology and framework for evaluating the ergonomics of UGI be established early, in tandem or prior to advancements in technology. The methodology utilised in this research is a solution to unsustainable development and provides a model of optimal UGI. Some of the methods and practices used by usability researchers and engineers have been explored in the creation of this model. An iterative framework will be developed so that the performance of users can be evaluated in parallel to the recognition accuracy of gesture lexicons. By understanding these issues the most appropriate uses of gesture can be defined and machine applications that maximise the use of gestures can be constructed.

### 2.2.4 Defining an approach for UGI development

Prior to developing open and accessible gesture interfaces, the capabilities and preferences of end users should be identified. Addressing these issues will help

identify whether a particular mode of interaction is accessible to heterogeneous users or only cater for people with a specialised need and ability. Through their research Nielsen et al. [2004] have recognised that interface developers generally prioritise computer recognition accuracy to the detriment of user comfort. Though some identify the conflicting priorities between developing fast accurate recognition and producing sustainable user interaction, few researchers present solutions to address this conflict. A user centric approach, which prioritises the physiological and cognitive concerns of people, is advocated in Nielsen et al. [2004] research. Though in the course of their research a number of gestures were evaluated, limited guidance was given about how to develop sustainable gesture vocabularies. Acknowledging that the optical recognition of a human-based gesture vocabulary presents significant technical challenges to interface developers Nielsen et al. [2004] suggests the development of shared datasets. He also suggests a set of guidelines for evaluating gestures. The resulting guideline uses five usability principles, defined by Nielsen [1994] , as a framework for assessing the performance of their gesture vocabulary (Figure 2.3 page 46). These principles evaluate how learnable, efficient and memorable an interface is. In addition, the potential errors and coverage a user encounters when using the interface is measured. The principles defined by Nielsen [1994] can be divided into two subsets. The first subset is concerned with how intuitively users find an interface; these principles include learn-ability, memorability and coverage. The second subset is concerned with the overall performance of an interface and measures the errors and efficiency encountered. For the purposes of this investigation, the second subset will be prioritised in the evaluation of gesture efficiency and user comfort.

As the user study undertaken in the course of this research is primarily concerned with identifying hand postures that can be comfortably replicated, the study focusses on evaluating perceived user preferences in addition to replication accuracy. The reason for this distinction is primarily regarding issues of identifying intuition, learnability and memorability beyond the context of a specific interaction. As a consequence the first subset of the principles defined by Nielsen [1994] will only be examined in the context of a specific interaction or interface.

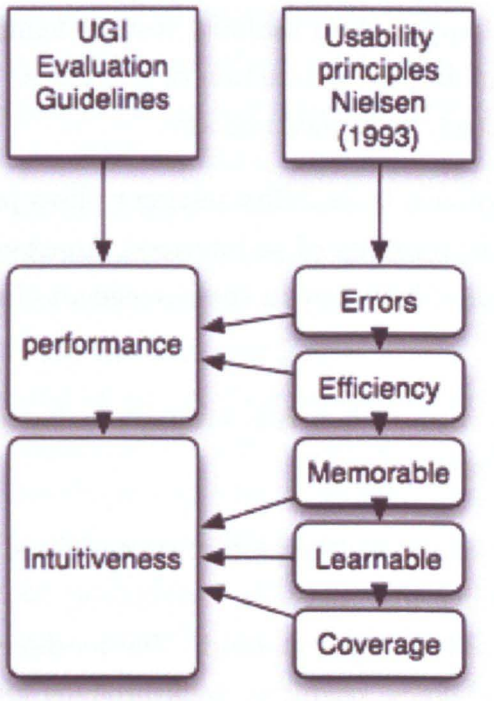



Figure 2.3: Categorising Nielsen [1994] five usability principles.

### 2.2.5 Usability testing methods

Computer interfaces have the potential to negatively impact human performance and affect human reliability, which can lead to significant errors occurring during human interaction [Coblentz, 1989; Rasmussen, 1987; Wiener, 1988; Woods et al., 1988]. There are a variety of different approaches that can be used in a usability test environment when examining the performance of an interface. The approaches most often used include the automatic, the empirical, formal and informal.
Usability testing methods:

- The automatic approach utilises a program or procedure to assess the performance of an interface. This may sometimes be in the form of a set of prescriptive guidelines that can be followed by inexperienced interface evaluators. .

- An empirical approach to usability testing defines the efficiency of an interface by examining the interaction of real users. This can be in the guise of a user trial study or a questionnaire.

- A formal approach to usability testing utilises precise models and formulas to measure the usability of an interface. An example of this approach is the popular usage of Fitts law in the assessment of cursor navigation of a GUI environment.

- The informal utilises heuristic evaluations based on the assessments of an experience evaluator.

In a study that compared the performance of some of the four most commonly utilised HCI usability test methods Jeffries and Desurvire [1992] identifies the benefits that derive from each method. The methods evaluated include heuristic, automatic guidelines, cognitive walkthroughs and the usability inspection method. Each of these methods where applied in the assessment of an interface. The interface investigated had two hundred and six predefined usability problems. The study identifies how successfully each method is able to identify inherent interface problems. From the two hundred and six known usability problems associated with the test interface heuristic analysis found 73 percent of them, usability inspection found 18.4 percent, guideline group found 18.4 percent and the cognitive walkthrough found 19.4 percent of the problems. In Jeffries and Desurvire [1992]study, heuristic evaluation was proven to be the best at finding the largest number of problems, these included those that were low priority together with a significant number of the most serious ones. Heuristic evaluation also proved to be the most expensive of the four techniques tested, and as a consequence might be unattractive to interface developers. The guidelines evaluation method and the cognitive walkthrough method where joint third. Both methods proved effective at finding general and recurring problems. Despite this, there were many serious problems that neither method managed to find. Jefferies research demonstrates that a well-designed set of guidelines can significantly aid novice evaluators or software developers to comprehensively examine the usability of an interface. The guideline evaluation method allowed evaluators to be

confident in their assessments of an interface. As it forced evaluators to engage in an extensive examination of the interface, as opposed to a narrow personal assessment. An evaluator using this method is significantly restricted to inspect the interface according to the guidelines. As a consequence an evaluators assessment can only be as good as the guidelines used. Despite this, there are genuine benefits that can be gain from using the guideline inspection method. The guideline inspection method enables non-usability specialist to assess the efficiency of an interface in the absence of a usability specialist.

Examining the potential of an interface prior to its development can be difficult and challenging. Usability specialists can be either expensive or hard to find, as a result of this expense the ergonomics of an interface is often evaluated late in the development cycle after substantive changes can be made. In Jeffries and Desurvire [1992] study no individual heuristic evaluator was able to find more than 40 usability problems. Notably, the number of problems identified by individual heuristic evaluators was similar to those found by guideline evaluators. The development of a robust set of general guidelines for inspecting the efficiency of gestures would greatly benefit UGI developers. The creation of an open iterative guideline framework that allows multiple evaluators to assess each others findings may produce an assessment as effective as heuristic evaluation.

## 2.3 Underpinning focus of methodology

UGI is based upon relatively novel technologies that are continually developing. Restricting the focus of interface research to the current limitations of hardware is not just short sighted, it is also unsustainable both economically and environmentally. Grounding UGI research on the capabilities of people would prove to be a more sustainable approach to interface development as the abilities of people are relatively constant. Such an approach would not only help to inform interface developers it would improve the durability of research outcomes. In the process of researching the implementation of unencumbered gesture interfaces, methods for evaluating the performance of users and machines have been undertaken. These methods are outlined in this section.

#### 2.3.0.1 Separating Syntax from the Interface

Once the capabilities of people have been prioritised, the need to separate a systems functional apparatus from a user's physical interactions becomes increasingly evident. At some stage in the evolution of unencumbered interface development a formal distinction between the action required to mediate an interaction and the mechanical system confronting the user will be needed. At such a stage it might be sensible to refer to the physical mechanism as the interface and the actions required to operate the system as the syntax. Emphasising these issues during interface development may limit the cycle of obsolescence to apply only to hardware not UGI syntax. This would also suggest that future evaluations would benefit from a having a broader focus beyond the proficiencies of the mechanical system.

#### 2.3.0.2 Methodological scope

It has generally been accepted that advancements in gesture-interaction will enhance user experience. However, there is still much potential for such interfaces to negatively impact human performance and reliability. Despite the recent developments in interface technology that include multi-touch and computer vision interaction, there is little understanding of how these interfaces will effect human performance during the operation of critical systems, such as those used in aviation, surgery and laboratories. A method for integrating and assessing the gestural preferences of users are presented in section 2.3.1 (page 50). The framework presented encompasses the initial stages of interface development, from its conception through to first prototype. The method is intended to encourage developers to assess and evaluate their proposed interfaces in the absence of a usability specialists, at a stage in the development cycle when substantive changes can still be easily made.

#### 2.3.0.3 UGI performance testing Issues

In UGI, the configuration of the interface is flexible and is currently not as well defined as with current hardware interfaces, such as the keyboard and mouse. Unencumbered gesture interfaces are of an augmented virtual and software na-

ture. In contrast to hardware interfaces software mechanisms are hidden from the user. For example, by pressing the keys on a keyboard a user will know whether the interface is responding to there input, through receiving either tactile, visual or audible feedback. The unencumbered interface will be void of any immediate tactile feedback and thus will have to rely on other forms of augmented feedback. Consequently the quality of user experience will be heavily reliant upon the speed and type of augmented response. As a result of these observations the underlying issue becomes a case of how best to study user performance and exertion. Subsequent methods will have to address more general aspects of user interaction, using anecdotal evidence to define user preferences and perceived physical exertion.

### 2.3.1 Methodology framework

There are relatively few guidelines available for evaluating the impact gesture lexicons have on people in comparison to those available for mechanical hardware interfaces. As a consequence of this limited resource this investigation intends to outline a methodology that could serve to enable a set of iterative guidelines to be created. Following what has previously been stated in section 2.3.0.2, the subsequent guidelines would only be intended to be utilised during the early phases of interface development when substantive changes can still be made and when the resources for a usability specialists are not available. Any subsequent guidelines should be open and iterative, so that they can be adapted universally. At present the method outlined is composed of twelve elements and is divided into three review phases.
Review phases:

- Design analysis
- Interface design
- Evaluation and comparison (see Figure 2.3 page 46).

#### 2.3.1.1 Design analysis

Develop performance evaluation method or use a pre-compiled efficiency dataset such as the GEf dataset (see Appendix 1.1 and 1.2).

Assessing physiological impact:

- Static load occurs when parts of the body are in one position for extended periods; this causes greater strain than when the body is active.
- Postures that deviate significantly from neutral posture are considered to be at risk for musculoskeletal stress. A neutral posture is where the joints are midway between full extension and flexion.
- Consider the environment where the interface will be used as movement and vibrations can change how effectively a gesture is replicated and recognised. This information will help to determine what types of gestures could be used in the operation of active and mobile interfaces.

#### 2.3.1.2 Interface design

In examining a framework consider how effectively it is capable of fulfilling these three criteria:

- Construct a coherent and unambiguous interface framework that limits the potential for negative feedback to be experienced by users.
- Construct an interface that either fulfils a needed requirement or is an improvement upon a given function.
- Evaluate the ergonomics and effectiveness of the underlying interface control framework.

#### 2.3.1.3 Evaluation and comparison

Consider how intuitively users find an interface, by examining these principle issues and evaluate the overall performance of the interface :

- Examine the repeatability of a lexicon
- Evaluate the effectiveness of a lexicon for completing specific tasks

- Evaluate how inclusive a lexicon is by finding the percentage of users able to successfully complete the specified action. This will help to determine whether it will benefit the majority of users.
- Examine human error
- Recognition accuracy
- Examine human-computer error ratio

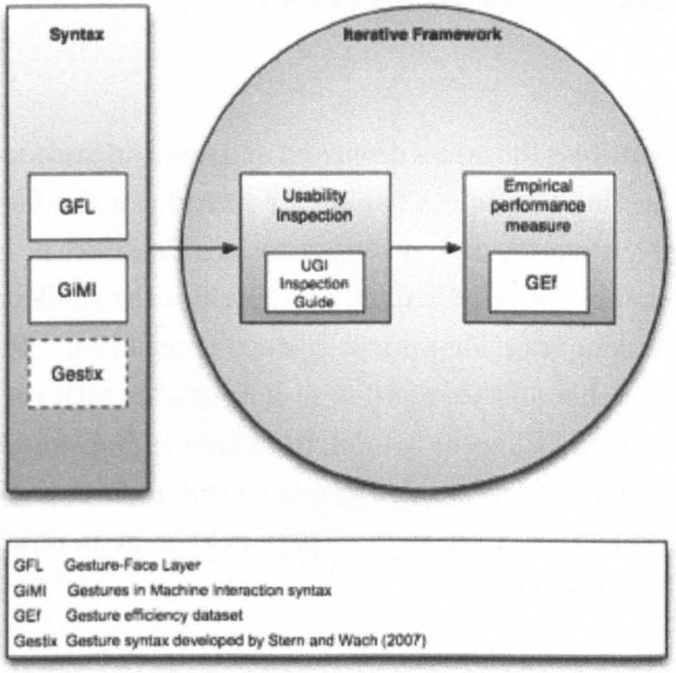



Figure 2.4: Illustrating the iterative process underpinning research method

#### 2.3.1.4 Guideline Application

The validity of these guidelines is documented in an empirical research investigation (Chapter 4, pages 74).

# Chapter 3

# Gesture Interface

Chapter 3.1 Outlines the issues discussed and gives an introduction to the research undertaken in this chapter. Chapter 3.2 introduces the Gesture-Face, which is a conceptual unencumbered gesture interface designed in conjunction with an optimised gesture lexicon. Chapter 3.3 describes the challenges confronted when configuring a computer for optical gesture recognition. Recommendations for improving detection and recognition in busy places with variable lighting conditions are presented. Chapter 3.4 describes how to implement 4D optical gesture recognition and discusses how this approach will improve ergotic, semiotic and deictic gesture detection. Chapter 3.5 Discusses the main outcomes of the research documented in this chapter.

## 3.1 UGI framework, developing the gesture-face

The purpose of this investigation is to demonstrate how to implement optical gesture recognition that is capable of accurately responding in real-time. An exploration of image processing and analysis has been undertaken, in order to achieve these aims. Efficient and robust solutions for real time optical recognition have been explored and will be presented. A prototypal interface called the gesture-face-layer (GFL) has also been designed. The techniques and processes used are able to facilitate robust interaction with machines especially when integrated with the GiMI syntax outlined in chapter 5 of this thesis. Applications

have been written in C++ and Matlab and components of these applications have been documented in appendix .3. This work demonstrates that accurate optical recognition of gesture is viable and can be implemented on a range of commercial computing platforms.

## 3.2 The Gesture-Face

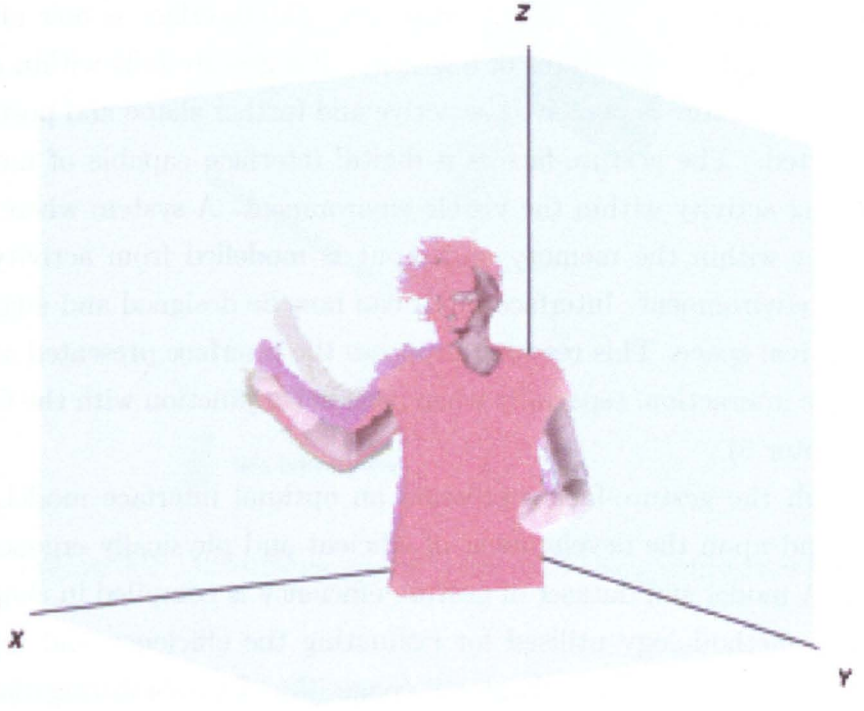



Figure 3.1: Illustrates motion and depth disparity model

In the development of the gesture-face layer a combination of techniques and methods have been utilised. Image segmentation has been conducted through the use of both motion and depth disparity mapping, Birchfield et al. [1999]. Shape and posture analysis is conducted through the use of statistical and probability modelling, using a combination of haar-like feature detection and boosting using

principal component analysis and mahalanbois distancing. The classifiers utilised in statistical training and posture estimation are built from motion templates and depth disparity maps (see appendix .3). The image maps produced enable the creation of an algorithm capable of estimating to a high level of accuracy the shape, form and context of a gesture. A gesture when performed by the user is captured and recognised through the use of motion segmentation, for code example see appendix .3 page 232. Once the region of interest has been determined, the proximity of the gesture is analysed using depth disparity mapping, for code example see appendix .3 page 224. The interface is now able to determine active and passive states of a gesture. If a gesture falls within a predefined distance the gesture is perceived as active and further shape and posture analysis is conducted. The gesture-face is a digital interface capable of modelling and recognising activity within the visible environment. A system where parameters can be set within the memory and input is modelled from activity in the 3D real-time environment. Interface input can now be designed and augmented into analogue real space. This research proposes the interface presented as a model of ergonomic interaction, especially when used in conjunction with the GiMI lexicon (see Chapter 5) .

Though the gesture-face represents an optimal interface model, its success will depend upon the development of efficient and physically ergonomic gesture lexicon. A model and dataset of gesture efficiency is compiled in chapter 4 (page 74). The methodology utilised for evaluating the efficiency and ergonomics of the dataset is discussed in chapter 2 (page 35). By monitoring the shape and position of the hand as it morphs through time we can begin to classify a broad array of gestures and human actions. Utilising all aspects of gesture in computer interaction presents an opportunity to move beyond the two-dimensional plane of current desktop interaction towards an interaction that is multi-dimensional. Through the combination of image processing techniques, such as 3D shape and motion detection, an interface capable of recognising the phases and conditions of gestures has been created. Using depth disparity mapping means that distant metric information can be used to define virtual spaces and surfaces. This information can subsequently be used to model virtual spaces that are sensitive to the interrelationship and proximity of people and objects. This method of spa-


Figure 3.2: Illustrates interactions with the gesture face layer

tial mapping is used in the creation of the gesture face layer. Creating a virtual surface in this manner enables touch screen like interaction to be implemented on to any open space or surface. The GiMI lexicon presented in chapter 5 (page 98) has been designed to work in conjunction with this mode of interaction. When the hand or finger breaches the surface of the gesture face layer the computer is programmed to monitor interaction in the same way that a mouse click symbolises user activity. The model of interaction that the gesture-face facilitates enables the combination of deictic, ergotic and semiotic actions to be utilised.

## 3.3 Image processing

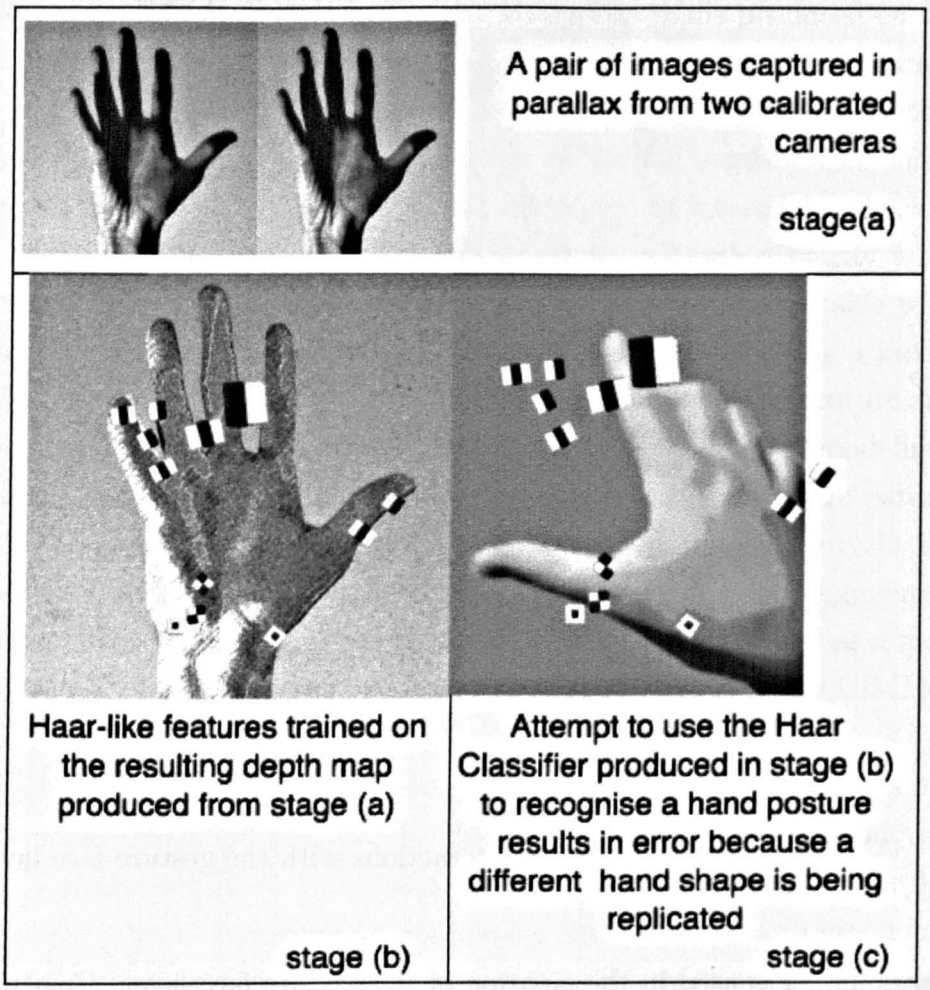



Figure 3.3: Illustrates the computational processes used by the GFL to facilitate gesture detection

### 3.3.1 Defining outlines

The first stage in this research centred on how best to enable an interface to optically recognise gesture. Before an object or form can be compared and contrasted against a range of possibilities, the interface must have the facility to distinguish between collections of forms. The first challenge in this process is to develop

an algorithm capable of separating the various elements of a visual scene. This process of distinguishing objects in an image is known as image segmentation and edge detection. Within computer-vision and image processing a variety of image segmentation methods have been developed and utilised, with varying degrees of success. During this investigation a range of methods have been explored, these include segmentation using colour, motion and depth disparities.

### 3.3.2 Colour segmentation

Colour segmentation has often been used for identifying the skin of people, in image analysis. This method can be useful for distinguishing a hand or a body part within an image. The advantage of this approach is that it makes training and detection quicker and simpler. However, there are factors that reduce the effectiveness of this method. The inherent variability of ambient light can considerably alter the appearance of colours. Moreover, the origin of a light source is an additional consideration when using colour segmentation. The position of light will determine the positions of shadows, potentially altering how objects are perceived. In addition, there is wide variation of skin tones that exist amongst the global population. The solution to skin-tone and light variation would be to create a dedicated process for monitoring light and skin colour variation. Implementing gesture recognition using colour segmentation can be effective especially using controlled light conditions. However, when light conditions are unknown or variable a greater amount of processing resources will be spent evaluating colour.

### 3.3.3 Motion segmentation

The ability to recognise motion allows us to see moving objects. Without this ability our perception and interaction with each other would be significantly hampered. People are rarely static, even when we sit still subtle movements betray our presence. To develop an interface capable of recognising gesture an algorithms for detecting and recognising motion is required. Though our physical gestures are three dimensional in form they operate within the fourth time dimension. When waving to greet a friend we typically move an outstretched palm from side to side. Without time as a variable there would be little to distinguish waving from

a request to wait or halt. Though we use static gesture in communication, the meaning and underlying intent of a gesture is evaluated in relation to the variable time. This corresponds to Neumann and Aloimonos [2000] theory regarding visual computation of physical action. Gestures are most often dynamic in nature, for example knocking on a door and clapping are dynamic. To recognise dynamic gestures we need to be aware of how they morph in time. Thus, time is a significant point of reference for determining the intent and consequences of a gestural action. Every gesture has a unique phase of time where it is active. Identifying these phases is an important element to consider when interpreting a gesture. Picking up a conversation in mid sentence will limit the overall comprehension of the discussion. This is also true with gesture recognition. As with all electronic input and output (I/O) communication the synchronisation of transmitted and received information is critical to successful interpretation of data. To be able to understand the meaning and intention of an action it helps to have perceived the action in its entirety. Determining where an action begins and ends is a significant aspect of gesture recognition. Algorithms programmed to recognise a gesture without reference to the time will consume extra processing resources through attempting to predict the phase and context of that gesture. Through utilising motion detection we can begin to isolate dynamic actions and plot a gestures path through time. The necessity to create artificial methods of synchronisation to dictate the tempo of interaction diminishes as a consequence. The creation of an algorithm capable of synchronising a computer to the various phases of gesture is vital for unambiguous recognition. By using a motion detection algorithm dynamic physical actions can be detected and recognised. Using this approach allows the phase of a gesture to be determined by the gesture alone. Detecting motion in a digital image sequence is a relatively simple process. Two images are captured at varying time intervals. The captured images are compared digitally to see whether there are any disparities between each image. The disparities identified illustrate motion within the image scene. These disparities can then be placed in separate image maps and an outline of the object in motion is revealed. Motion detection is a simple and effective method for distinguishing dynamic human actions from static environments. This makes motion detection a valuable technique to use in image segmentation. Using motion detection in

image segmentation resolves the problem originally encountered by early gesture recognition systems where sensitivity to the variability of light, background scenes and a users clothing could cause significant recognition problem. The use of motion detection in segmentation creates an algorithm able to recognise the active element within a scene, in this case gesture. The moment a gesture becomes active the outline and shape of that gesture can be clearly determined. An example of segmentation using motion detection can be seen in figure 3.4.

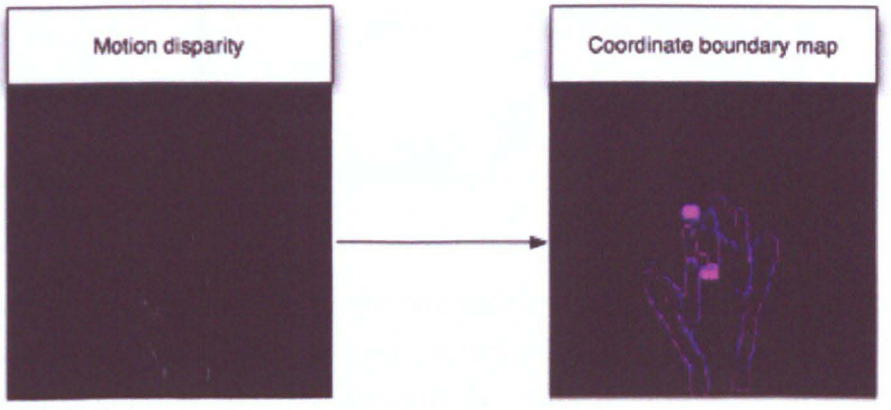



Figure 3.4: Image map of motion disparity map

This method can be expanded to produce a template of human action, where the passage of time can be represented in a single image map. Rather than simply analyse a single instant of motion to determine a fragment of a gesture, a gesture can be mapped to illustrate the course of an action from its beginning to its end. The image map created enables the entire phase of a gesture to be visually classified as a single image template. The motion history of objects can be represented in a single time frame. As a result physical actions can be classified and interfaces can be programmed to visually recognise gesture.

Bobick et al. [2001] together with Weinland et al. [2006] utilise similar techniques for capturing and representing human action in a 2D image map. The resulting images can be processed using pattern recognition techniques such as haar-like feature detection and principal component analysis (section 3.4.1.2 page 70). Optical interfaces can be trained to recognise motion histories in the same way that they are trained for object recognition. This ability to produce and

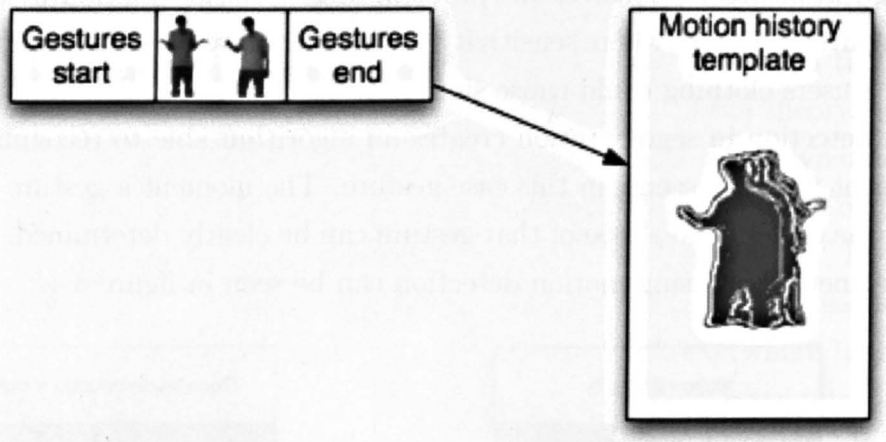



Figure 3.5: shows an example of a motion history image map

record motion histories means that the phase of a gesture can be digitally analysed and a dataset of gesture can be produced. The implementation of these methods has been documented in Appendix .3, for working code example see page 232.

## 3.4 Defining the gesture-space

#### 3.4.0.1 The range of gesture

Our gestures provide us with access to multiple spaces. Reaching either up or down we can access objects and spaces that are proximal to us. The same is true for accessing north, south, east and west spaces. Gestures allow us to project our intention into the fourth dimension time. For example, bowling with topspin at an opponents cricket stumps or slamming a door demonstrate gestures that exhibit delayed consequences. Gestures also allow us to sense and feel elements of our physical environment, providing people with access to a sensory tactile space. Through physical contact our gestures introduce the physical qualities of objects, such as the texture of a surface. There are aspects of gesture that enable us to survey the physical composition of object and environments. Through the application of force our gestures are also able to convey information about the

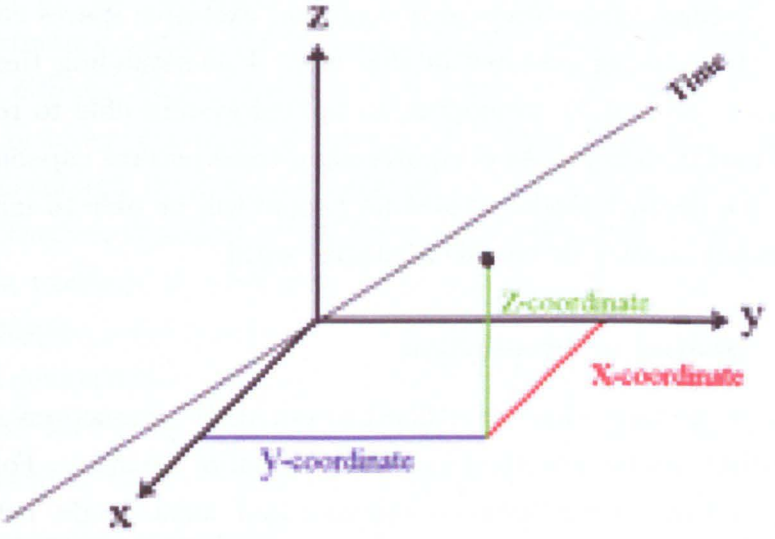



Figure 3.6: Illustrating the dimension of gesture

physical composition, such as the weight or physical density, of objects. In this research a digital interface capable of translating these six aspects of gesture has been created. The most significant of these is equipping machines with the ability to interpret physical force through vision alone.
The axis of gesture:

- North and South
- East and West
- Up and Down
- Now and Later
- Tactile sensory feedback and Exertion of physical force
- Static or Active

The individual spatial characteristics of gesture provide another tool for recognising a gestures type. Maps of mutual and exclusive spaces can be created as a result of analysing gestures in this way. Understanding these spatial characteristics is critical to developing an optical system able to recognise the full range of human actions. As computer algorithms become capable of differentiating between distinct modes of gesture people will be able to manipulate digital environments as they do the 3D analogue world.

#### 3.4.0.2 Spatial differentiation

The diverse gestures that we utilised in our daily interactions occur in regions of space that can be described as either mutual or exclusive. For example, most semiotic gestures occur between shoulder and waist height between forty and twenty centimetres from the torso. Gestures that adhere to these parameters occupy a mutual space that is shared. An outstretched finger on a fully extended arm is the furthest point an individual can reach using the upper limbs. Consequently, the deictic pointing gesture is an example of a gesture that can occupy a region of space that is exclusive. When performed by a typical adult, the pointing gesture extends between forty and fifty centimetres beyond the torso. Provided that every gesture is measured from the same relative positions, such as the head or torso, the mutuality and exclusivity of a gesture can be measured.

#### 3.4.0.3 Depth segmentation

Depth recognition is an important element of human perception. It equips people with the ability to perceive perspective, distances and speed. Perspective is a prerequisite for the visual recognition of three dimensional space and objects. Stereovision allows people to perceive depth and perspective. Without binocular vision our ability to see objects in space is significantly hampered. Binocular vision can be easily implemented on an interface by the use of two image sensors. In this study two web cameras with wide-angle lenses were used. These lenses were used in order that the whole upper torso of a user can be observed during interaction. Birchfield et al. [1999] creates an algorithm for combining a pair of images, taken from parallel viewpoints, into a single binocular image map.

Table 3.1 demonstrates the implementation of Birchfields algorithm, using the opencv image-processing library. The Birchfield et al. [1999] algorithm can also be used for creating motion disparity maps. Except instead of comparing images captured at different time intervals, images taken from parallel viewpoints are compared. In order to obtain the best results using Birchfields algorithm the binocular viewpoints needs to be calibrated. Calibrating the two cameras is necessary if there is significant radial distortion produced by the lenses. Figure 3.7 shows an example of a disparity map created from a pair of uncalibrated cameras. Despite being products of uncalibrated stereo images the disparity map created demonstrates how depth perception can be successfully implemented in computers. Metric information can be extracted from the three-dimensional scenes produced through stereo disparity maps.

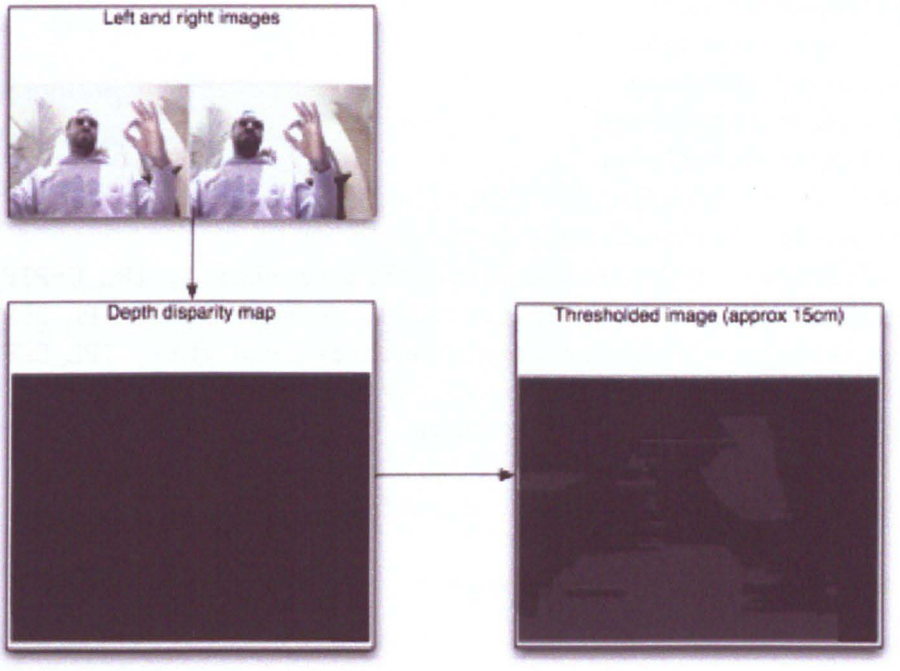



Figure 3.7: Image map of depth disparity from uncalibrated stereo images

The proximity of objects can be distinguished from the shade of pixels defining the object. The lighter the pixels the closer the object, while the darker the pixel the more peripheral and distant the object. While the successful implementation

Table 3.1: Demonstrates the stereo disparity using Birchfield et al. [1999] algorithm

```
// Stereo_disparity.cpp

IplImage* srcLeft
IplImage* srcRight
IplImage* leftImage
IplImage* rightImage
IplImage* depthImage
 srcLeft = cvLoadImage("right.jpg",1);
 srcRight = cvLoadImage("left.jpg",1);
 leftImage = cvCreateImage(cvGetSize(srcLeft), IPL_DEPTH_8U, 1);
 rightImage = cvCreateImage(cvGetSize(srcRight), IPL_DEPTH_8U, 1);
 depthImage = cvCreateImage(cvGetSize(srcRight), IPL_DEPTH_8U, 1);
cvCvtColor(srcLeft, leftImage, CV_BGR2GRAY);
cvCvtColor(srcRight, rightImage, CV_BGR2GRAY);
cvFindStereoCorrespondence( leftImage, rightImage,
  CV_DISPARITY_BIRCHFIELD, depthImage, 50, 15, 3, 6, 8, 15 );

cvShowImage("disp",depthImage );
```

of depth recognition enables the proximity of different objects to be recognised it also facilitates the perception of three-dimensional objects. As a consequence, a greater amount of detail regarding the shape and form of a gesture can be obtained. Such techniques have the potential to improve the accuracy of hand posture recognition. The effectiveness of these methods has been analysed by Munoz-Salinas et al. [2008]. The methods utilised in this investigation have been found to be some of the most robust. Investigating whether depth recognition affects the accuracy of algorithms such as PCA and exemplar detection Munoz-Salinas demonstrates that each method benefits significantly from the use of depth silhouettes.

### 3.4.1 Statistical modelling

#### 3.4.1.1 Haar-like features

Utilising motion detection and depth templates a robust and efficient method of segmentation has been achieved. Since the hand can now be successfully distinguished from the ambient environment, the process of object recognition can begin. Various combinations of algorithms have been used in this study to enable recognition of specific gesture and postures. The preferred method is to use a training algorithm using haar-like feature detection in combination with principal component analysis and mahalanbois distancing. In the first stage of this process, the computer is trained to distinguish the hand in an image using haar-like feature detection, Viola and Jones [2001]. Haar-like features comprise a series of two-dimensional shapes of varying orientation and patterns subdivided into various black and white patterns ( see figure 3.8). Through overlaying these features over the source image and calculating the combined pixel intensities within these regions the difference between the features and source can be calculated. A feature can be place at any location within the source image. Some features work better for detecting certain images, depending on the feature pattern. The advantage of haar-like feature is its calculation speed. However using a single feature will facilitate detection marginally more than fifty percent of the time, which is considered statistically little better than random. As a consequence of the low recognition accuracy multiple feature are used simultaneously in a process

called haar classifier cascades.

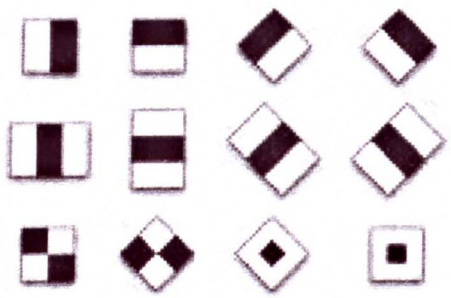


Figure 3.8: Haar-like features

Statistical training using haar-like features is achieved by digitally processing two separate image directories. One directory contains images that depict the object to be recognised against an array of diverse backgrounds. The other directory contains images of the backgrounds with no reference to the desired object. The algorithm selects the haar-like features that best reflect distinctive characteristics that identify the chosen object. In this investigation haar-like features cascade has been trained using Opencv. After training these features on a dataset of hand postures an xml file is produced. This file documents how each individual haar feature conforms to each posture.

The feature detection file can then be incorporated into a detection algorithm and the presence of a hand posture can be predicted through the analysis of image maps. Once the training process has been completed classifiers representing the object are produced. Classifiers created for the purpose of hand detection are consistently able to identify the hand. However, when using this method a significant amount of false positives can be observed. In addition to this problem, training computers for optical recognition using haar-like features requires large datasets of images. Assembling large datasets is time consuming and can subsequently slow the development process.

There are ways to reduce the time it takes to compile the training set. For example generating multiple images using chroma-key [Anton-Canalis et al., 2005]. This entails superimposing a diverse range of backgrounds behind the desired object to create thousands of examples of the object in a variety of environ-

Table 3.2: shows a sample of haar-like feature cascade xml file trained to detect the hand

```
// mono_20_hand.xml
<feature>
            <rects>
               <_>
                 8 3 12 5 -1.</_>
               <_>
                 12 7 4 5 3.</_></rects>
             <tilted>1</tilted></feature>
           <threshold>-0.1828908026218414</threshold>
           <left_val>0.7676910758018494</left_val>
           <right_val>-0.8145673274993897</right_val></_></_>
       <_>
```

Table 3.3: shows the hand detection cascade being utilised in an algorithm (for full example see appendix .3 page 208

```
// gesture_detect.cpp



cascade_name = "mono_20_hand.xml";
    cascade = (CvHaarClassifierCascade*)cvLoad(cascade_name, 0, 0, 0);
if( cascade )
    {   CvSeq* faces =
cvHaarDetectObjects(img, cascade, storage, 1.2, 1, 0, cvSize(24, 20));

       for( i = 0; i < (gesture ? gesture->total : 0); i++ )
        { CvRect* r = (CvRect*)cvGetSeqElem( gesture, i );
```

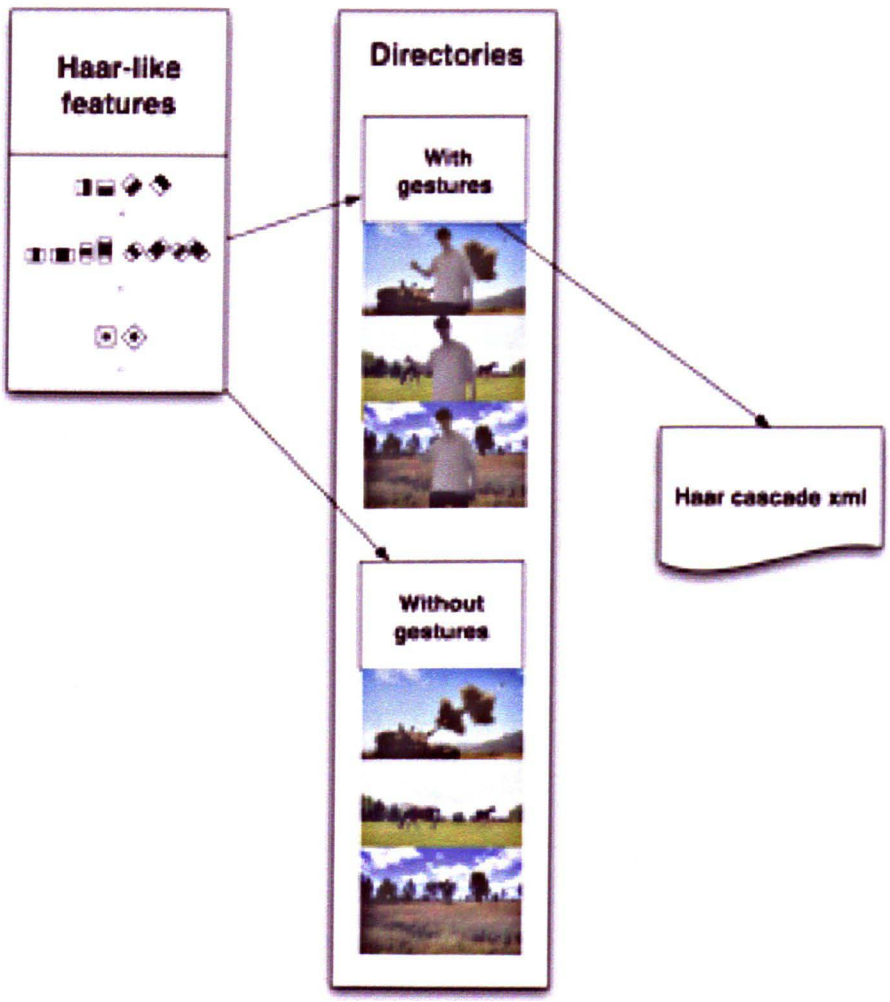



Figure 3.9: Training interfaces using haar-like features

ments and conditions (see figure 3.9). These methods enable the quick assembly of large datasets on which haar-like features can be trained. This is a process that can be easily automated. Anton-Canalis et al. [2005] also demonstrate that the accuracy of the training set does not diminished as a result of using this method. The classifiers produced using haar-like features though effective when used to detect a generic hand shape are not sufficient to detect subtle variation in a postures shape. The classifiers produced are best suited to being weak classifiers for detecting the generic hand. After identifying a hand using these weak

classifiers detection can be boosted using Principal Component Analysis (PCA), Independent Component Analysis (ICA) or Linear Discriminant Analysis (LDA). In this investigation PCA has successfully been utilised for boosting the statistical accuracy of gesture detection. In chapter 4 the effectiveness of this method is demonstrated during the compilation of psv index in the GEf dataset. Figure 3.11 shows a screenshot from a prototype developed during this investigation, which has been trained using haar-like features and boosted using principal component analysis. Research has proven that principal component analysis can significantly boost accuracy of classifiers derived from haar-like features [Zhang et al., 2004].

#### 3.4.1.2 Principle component analysis

PCA is a mathematical function that allows multi dimensional arrays, such as digital images, to be reduced into eigenvectors. The vectors produced can be used to determine similarities and variations between images. In this research PCA analysis has been conducted using MATLAB image processing tool kit and OpenCV. The application and implementation of these methods are presented in Appendix .3. The steps undertaken when calculating the principle components of data are as follows. First, we need to identify the relevant data to analyse, in the case of this research this data is a dataset of images. Second, we need to subtract the mean of the dataset from each piece of data. For example when calculating the PCA of an image matrix you would subtract the mean of all pixels on each row from each pixel value within that row. Third we need to calculate the covariance between each row within the matrix using the formula shown in figure 3.10. The aim at this stage is to examine the relationship between the various dimensions. Fourth, the data derived during the calculation of our covariance matrix provides us with our eigenvectors and eigenvalues. An eigenvector is a vector that has been scale up during matrix transformation and the eigenvalue is the amount to which the eigenvector has been scaled. Finally, we sequentially arrange the data so the that eigenvectors with the highest eigenvalues form the first principle components of our data. Remember the main purpose of this process is simply to reduce data of higher dimensions into lower dimensions so that any underlying patterns are easier to identify, in otherwise noisy datasets. A PCA calculation

$$C = \begin{pmatrix} cov_{x,x} & cov_{x,y} & cov_{x,z} \\ cov_{y,x} & cov_{y,y} & cov_{y,z} \\ cov_{z,x} & cov_{z,y} & cov_{z,z} \end{pmatrix}$$

Figure 3.10: formula for calculating the covariance of a matrix

in of itself does not provide any answers it simply enables us to plot data and examine patterns for trends or disparities.

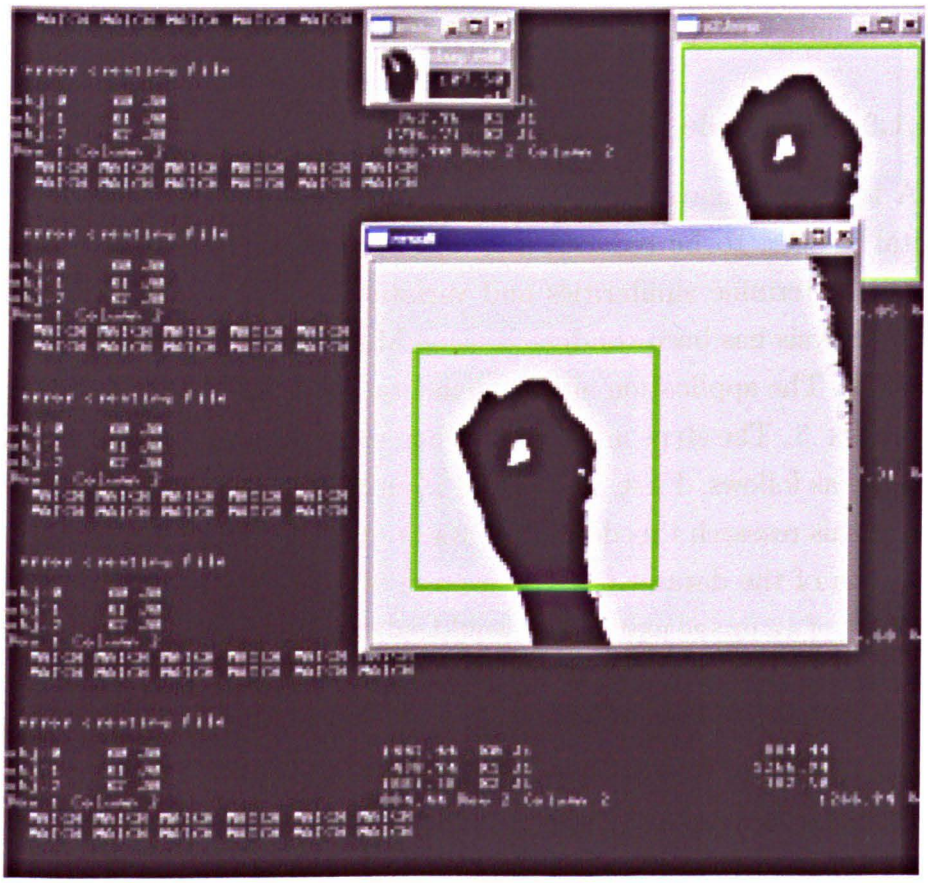


Figure 3.11: Screenshot of symbolic gesture recognition

## 3.5 Discussion

The work discussed in this chapter underpins the practical inquiry behind the whole of this thesis. The outcomes of this chapter have help to frame the concerns and issues driving this investigation into the feasibility of using unencumbered gesture for machine interaction. However though some aspects of the work described might seem technical it should not distract from the creative motivation underlying this work. The field of image processing maybe perceived as a domain for computer scientist and mathematicians. However, the contributions artists and creative practitioners make to the broader discourse of how we think about and interpret the visible environment should also be considered. Furthermore, the approaches creative practitioners utilise when confronting obstacles in their practical work is also a significant factor in the way this research has been approached. Through the process of documenting technical aspects of implementing an optical interface the creative practice of trial and error leading ultimately towards discovery might not have been sufficiently illustrated. However through the process of this research the computer-vision and image-processing medium has been explored in a similar vein as a painter might organise and mix paints, decide the material for the canvas and select a range of paint brushes. Along the way traditions and conventions may be followed but only through continually engaging with the creative practice can insight and innovation be achieved.

Through a process of playing with the medium of image-processing to see what works, this research identified significant aspects regarding the visual recognition of active people. For example the six axis of gesture discussed in section 3.4.0.1 though they might be obvious represent a consideration of how people physically inhabit space and lead to deeper questions about what constitutes an action. Such consideration do not simply represent a technical inquiry into image processing. Such inquiries are weighted more towards a contemplation of vision, perception and actions. Hopefully as a result of this process other practitioners will find it helpful to consider the task of optical gesture recognition as being more than simply a technical issue. Beyond being purely a technical investigation this is an inquiry into the process of perceiving the world around us.

Though some of the techniques and approaches used in this inquiry may not

be completely original, the reasons for there implementation was because a need was identified. In the case of using mahalanobis distancing to identify similarities in a dataset; it is well established within mathematics that this formula can be used in such cases. However understanding the results of a calculation is distinct from performing the calculations on complex data. In the course of this research investigation a wide range of calculations have been understood and performed.

Though much has been learnt as a result of the many problems encountered, the most significant outcome of the research undertaken in this chapter is the design and creation of the gesture-face layer. It is an original solution to the problem of determining when a participant is actively engaging with an optical interface, which is by nature immersive. Once you are in the gaze of the camera lens you become active within an augmented real data-space. Under these circumstances it would be useful to have a mechanism for expressing a desire to participate in such a space. Further to addressing this issue the gesture-face layer offers additional flexibility of allowing users the ability to also define the position and size of the interface. Any surface within perceptual range of the optical interface could be defined as an input interface. Though the concept of being able to superimpose control functions of electronic devices on to real objects has parallel with Fails and Jr. [2002] LightWidget, the accompanying gesture lexicon enables the interface to be re-orientated in realtime.

# Chapter 4

# Evaluating gestures

Chapter 4.1 examines the physical preferences of gesture interface users whilst contrasting the performance of user interaction, this study investigates the efficiencies of a set of symbolic gestures by conducting a user study . Chapter 4.2 presents an analysis of the data compiled in this study and discusses the accuracy of subsequent findings. A detailed interpretation of these findings is also presented.

## 4.1 User study to evaluate gesture efficiency (GEf)

The emergence of novel gesture recognition systems presents developers with the opportunity to redesign the human computer interface. In order to help prevent unsustainable paths of development where unnecessary physical and cognitive loads are placed upon users, this investigation into how comfortably people perform gestures is undertaken. By the use of a questionnaire and interview this study measures how comfortably and accurately a user sample replicate a set of symbolic hand postures.

### 4.1.1 Methodology

It could be argued that comfort is a purely subjective notion and what is considered comfortable by one individual should not be generalised across a broad sample. However, in order for an interface to be ergonomic and of benefit to the

broadest range of users, a method for generalising the idea of comfort needs to be defined. During the design of this user study a range of methods for testing comfort have been investigated. Within clinical and bio-mechanical practice the range of methods often used are electromyography (EMG) and Surface electromyography (SEMG). These methods measure muscle activity by monitoring the small electrical charges that are released by muscles when they are active. However, measuring physical exertion using EMG is very invasive, as this method requires needle electrodes to be inserted into the muscle. As a result of this process only a finite number of muscles can be tested at any one time. Though surface electromyography (SEMG) can be used as a less invasive alternative, SEMGs can only provide a limited picture about exertion experienced by the human hand. Neither of these methods represent a practical solution for this user study as access to equipment and expertise in this area is very limited. Instead of collecting data with EMG or SEMG, which would still require anecdotal user input to qualify any results, a questionnaire based data collection method has been adopted. This alternative questionnaires based approach has been demonstrated to be a viable alternative for extracting data regarding perceived physical exertion from test participants [Sommerich et al., 1996].

#### 4.1.1.1 Data collection method

During the course of this user study into gesture comfort, a questionnaire will be used as a method of collecting participants perceptions of exertion. In the questionnaire participants are asked to replicate a set of hand postures as accurately as possible and rate each posture according to how comfortable they perceive them to be. After a process of analysis (see section 4.1.4) the ease with which participants replicate a set of actions will be collated in a gesture efficiency dataset (GEf) (See Appendix .1.1, pages 156) for the primary purpose of classifying how comfortably and efficiently people perform a set of gestures. Though this study is not a comprehensive study of all gesture the resulting dataset should enable gesture interface researchers to identify the underlying ergonomics of a specific set of hand postures (See Appendix 1, pages 156). The conclusions of this study should also inform unencumbered gesture interface developers about the feasibility of

employing a questionnaire to assess the ergonomics of a gesture lexicon.

#### 4.1.1.2 Hypothesis 0

The null hypothesis to this study states that using a questionnaire to catalogue and classify the amount of physical exertion experienced by individuals within a sample group will fail to find common relationships within the data, unless purely by chance. Subsequent outcomes of analysis could not be used to differentiate comfortable gestures from uncomfortable ones (See formula 4.5).

$$H0 : S \leq 7 \vee \geq 2 \tag{4.1}$$

#### 4.1.1.3 Hypothesis 1

The alternative to the null hypothesis is that questionnaires can be effective methods for producing generalised models of perceived user exertion. The method employed is robust enough to facilitate the creation of a coherent gesture comfort index (See formula 4.6).

$$H1 : S > 7 \ \& \ \sigma < 2 \tag{4.2}$$


Figure 4.1: Lexicon of hand posture tested during user study

### 4.1.2 Test conditions

Over the course of one day students and members of staff attending Camberwell College of Arts were asked at random to participate in a user study examining gesture recognition and user comfort. When asked whether they were interested in participating in this study (82.6 percent) showed enthusiasm for this concept, few were uninterested. After consenting to take part in this study participants were taken to a quiet corner of the student union cafe and ask to sit in front of a computer terminal. They were then ask to respond to a paper based questionnaire. The questionnaire consisted of three A4 sheets of paper with images representing fifty-two hand postures (See Appendix .1.1). Once seated participants were then asked to replicate each hand posture in sequence, whilst facing a computer web camera (See figure 4.2). During the process of replicating each posture their image is captured via the computer web camera and stored for future image analysis. After replicating each posture participants were asked to give a rating of between 0 and 10, regarding the amount of exertion each action required. Each score and remark was documented by the test supervisor to allow participants to focus purely on replicating each action.

### 4.1.3 Participant details

The age range of the candidate pool was between eighteen and thirty. Those willing to participate in this study had ages ranging from between eighteen and twenty-six. The gender distribution was that of eight (42.1 percent) male and eleven (57.8 percent) female. Two (10.5 percent) of the participants stated that they suffered from physical conditions that might restrict their hand movements.

### 4.1.4 Procedure

The responses of participants will be used to assess the levels of perceived physical exertion they experience. The total score of each posture will be then aggregated and the mean of the results are calculated. In order to evaluate whether the mean accurately represents consensus within the user sample the standard deviation is calculated, using a formula specific to calculating variation within a sample not

Figure 4.2: Participant in user study

a population (See formula 4.3). Once the mean physical exertion score has been calculated together with its standard deviation a priori set of bench marks will be defined. After the data has been compiled in this way there significance will be evaluated using a Student T test to determine whether such an approach is robust enough to be an effective predictor of exertion and comfort. In section 4.2 (page 85) after the significance of the results have been calculated a posteriori set of conclusions will be drawn.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}} \tag{4.3}$$

Images of participants replicating predefined actions are captured. Silhouettes

representing each replicated posture are then produced (see figure 4.3). Prominent features such as the index or middle fingers are then roughly aligned and the images are analysed using principal component analysis (PCA) and mahalanobis distancing (See Appendix .3.2.1).

### 4.1.5 Classification

A participants assessment pitched at the lower end of the scale will constitute a negative appraisal of a posture, suggesting discomfort was experienced. An assessment pitched towards the top of the scale will constitute a positive appraisal of a posture, suggesting comfort was felt. An assessment of five will constitute a neutral assessment of a gesture and values of seven indicate the lowest positive assessment this study will accept as representing comfort.

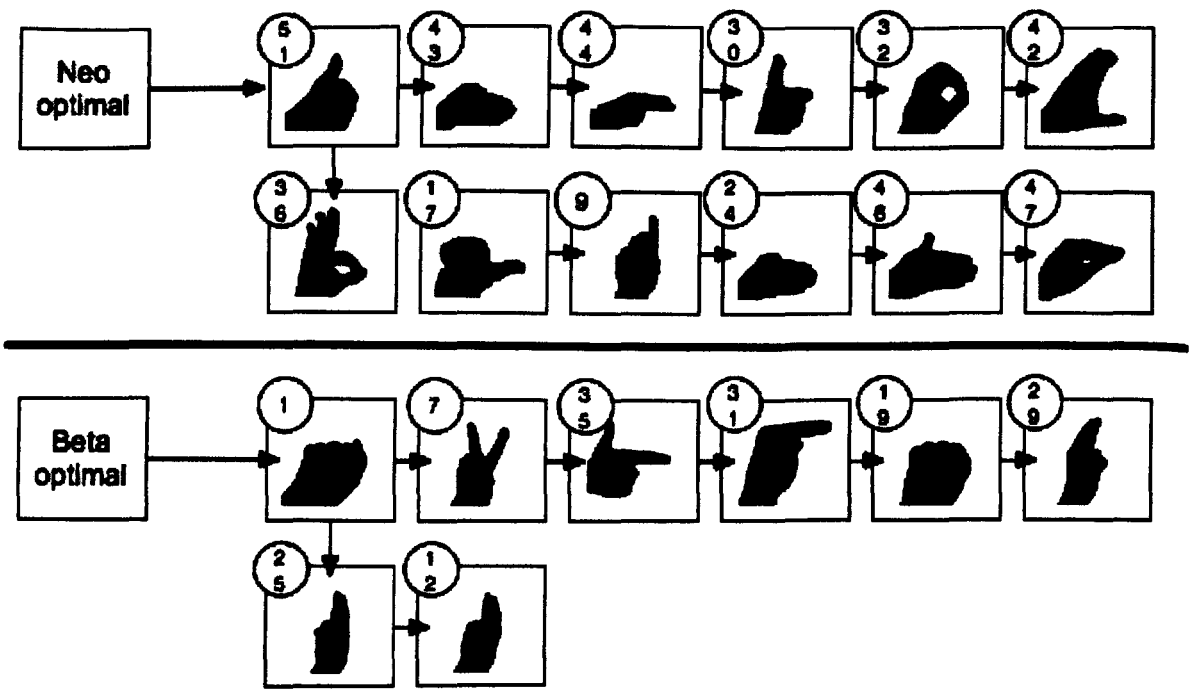



Figure 4.3: Order of perceived comfort for neo and beta-optimal postures.

After calculating the mean and standard deviation of results collected an initial appraisal of participants PPE scores will be conducted. After this process

a posteriori set of conclusions will be drawn once a statistical students T test has been conducted. This has been done in order to test whether the use of a questionnaire in combination with clear direct benchmarks can be used to both identify user preference and determine the boundaries of user comfort. The outcomes of the benchmarks defined below will be justified if they reflect or correlate with trends identified in a Students T tests. Provided there is a correlation between results this method should be used to encourage unencumbered interface developers to conduct similar tests when developing interface syntax or gesture lexicons, even in the absence of a usability specialist.

#### 4.1.5.1 A priori set of benchmarks

The overall mean and standard deviation of each posture has been calculated and can be seen in table 4 (Appendix .1). The results derived from the application of these measures fall into five distinct categories. These categories are defined as the optimal-comfort threshold; the meta-comfort band; the neo-optimal comfort threshold; the beta-comfort threshold and the sub-optimal threshold.
Five categories:

- First Category: Optimal comfort threshold
  - Consensus two standard deviation from the mean
  - Mean comfort preference of eight or above
- Second Category: Meta-optimal comfort threshold
  - Marginally greater than two standard deviation from the mean
  - A mean marginally less than eight
- Third Category: Neo-optimal comfort threshold
  - Absolute and borderline consensus two standard deviation
  - A mean marginally less than eight or above
- Fourth Category: Beta-optimal comfort threshold

  - Absolute and borderline consensus two standard deviation
  - Mean comfort preference of between seven and eight

- Fifth Category: Sub-optimal
  - Outside consensus threshold
  - Mean comfort preference of between one and seven

#### 4.1.5.2 The optimal-comfort threshold

The first subset contained within the PPE index is the optimal-comfort threshold. This threshold identifies postures that can comfortably be utilised in computer interaction. There are two sets of criteria that have to be fulfilled for a posture to be classified as optimal. First, the average assessment of a posture should be a score of eight or above. Second, a consensus of opinion should be reflected within the overall set of results. As small variations suggest consensus the optimal-comfort threshold only includes postures that exhibit a low level of variation across the sample. Hand postures calculated to have standard deviation of less than two represent postures that reflect consensus. The parameters of the optimal-comfort threshold are represented by the mean eight and above together with a standard deviation of two or less. In figure 4.4, the postures that adhere to these parameters are located above the solid horizontal line extending from eight on the y-axis and to the left of the solid vertical line extending from two on the x-axis. This threshold has been used to identify gestures that could be appropriately utilised in sustainable interaction.

#### 4.1.5.3 The meta-comfort band

The second subset of the PPE index is the meta-comfort band. The postures defined by this threshold represent gestures that fall marginally outside of the optimal-comfort threshold. For example, there are two postures that fall below the mean comfort measure eight by less than six hundreds of a decimal place. Furthermore, there is one posture that is less than six thousandths of a decimal place beyond the threshold delineating the optimal range of variation and consensus. These three postures can be seen in figure 4.4, the threshold representing

these postures correspond to the dashed line. This threshold has been created in order to provide a degree of flexibility when assessing borderline gestures.

#### 4.1.5.4 The neo-optimal comfort threshold

The third subset of the ppe index is the neo-optimal threshold. This subset of postures includes postures that fall within the optimal-comfort threshold and the meta-threshold band. These postures will be evaluated to see the degree of shape variation that exists within this set. These results are documented in the psv index (See Appendix .2).

#### 4.1.5.5 The beta-comfort threshold

The fourth subset of the PPE index is referred to as the beta-comfort threshold. This threshold identifies those postures that exhibit either absolute or borderline consensus and a mean comfort score of between seven and eight. This threshold includes postures that with further testing might fulfil the criteria of the optimal-comfort threshold; after either recompiling the dataset with alternative statistical models or using a sample large enough to provide generalised conclusions. As a consequence these postures will also be evaluated to see how much shape variation can be observed in the replicated postures of study participants.

#### 4.1.5.6 The sub-optimal threshold

The fifth subset of postures in the PPE index characterise those that do not show consensus or do not have a mean that reflects a positive assessment of a posture. These postures have been deemed as either uncomfortable or inconclusive.

### 4.1.6 Threshold distribution

Of the fifty-two postures examined in this user test only nine (17.3 percent) have been identified as belonging to the optimal-comfort threshold (figure 4.4). Three postures (5.8 percent) belong to the meta-comfort band. Twelve (23 percent) postures fall within the neo-optimal threshold. There are eight (15.4 percent) postures belonging to the beta-optimal threshold and all other hand postures

(59.8 percent) are sub-optimal. Twenty postures (38.4percent) have been analysed and documented in the psv index. These postures include those represented in the neo, meta and beta-optimal comfort thresholds.

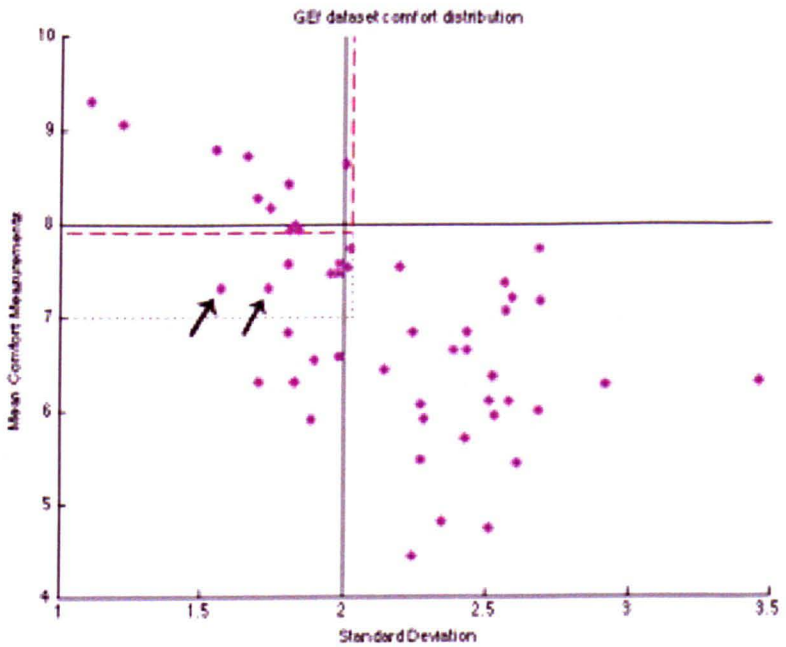



Figure 4.4: Graph representing the mean comfort measure and Standard deviation of each hand posture

There are two alternative representations of the same posture included in the GEf dataset. One representation is categorised as the Japanese manual letter U, posture number twenty-five in the GEf dataset sequence (See Figure 4.1, page 76). The other posture is categorised as the American U, posture number twelve in the GEf dataset sequence. Notably, these two postures received an identical mean comfort measure and exhibit similarly high degrees of consensus amongst participants assessments. The proximity of these two assessments can be seen in figure 4.4, the positions of these postures are indicated with two arrows. Analysis of the dataset shows that certain gestures are generally perceived to be more comfortable than others. The analysis also shows that participants perform some gestures more uniformly than others. The postures identified in the neo-optimal

and beta-optimal threshold are postures that have been identified as potentially the most sustainable. However, before these postures can be defined as suitable for use in computer interaction there are further sets of criteria that have to be met.

#### 4.1.6.1 Defining accuracy and errors

An interfaces efficiency is generally determined by the stability of a users performance. The efficiency of an interface should also be measured according to the range of errors encountered. In the context of this study a gestures overall performance will be measured in two ways. First, the level of shape variation observed when participants replicated each posture will be analysed. Second, the range of errors users encounter will be evaluated. The errors that are evaluated will consist of unregistered postures that have not been successfully captured either as a result of being performed off camera or the orientation of the posture is inconsistent with the images represented in the questionnaire.

#### 4.1.6.2 Measuring uniformity of posture replication

The first phase in this process is concerned with measuring how uniformly participants replicate each posture. Determining which postures are predisposed to shape uniformity is a crucial element in developing an interface capable of sustaining interaction with a diverse range of people. Quantifying the proportion of shape uniformity intrinsic to each posture enables consistently recognisable gestures and postures to be identified. Furthermore, establishing the level of shape uniformity will also define the likely stability of each postures performance. In this study posture shape uniformity has been calculated using the Matlab and OpenCV statistics and image libraries. Through using this software digital images can easily be converted from pixel values into simple number vectors, arrays and matrices. Once an image has been converted in this way a range of statistical processes can be deployed. In the case of this study principal component analysis and mahalanobis distancing were used. For an example of the algorithms needed when performing these operations see appendix .3.2.1 and appendix .4.3.1 on pages 180 and 219 , respectively. For a more detailed explanation of how PCA

and mahalanobis distance work see Chapter 3.4.1.2 on page 70.

### 4.1.7 Accuracy of posture replication

The second phase of evaluation defines how accurately each posture was replicated by the sample. Defining the accuracy of performance is significant to understanding the general tendencies of the sample. For example, all participants may consistently make the same errors when replicating a posture. Alternatively, postures that utilise specific fingers may produce a higher percentage of errors than others.

## 4.2 Interpretation of analysis

The results of this analysis have been plotted on scatter graphs producing a set of twenty distinct cluster groups (See Appendix .2). Each cluster consists of twelve individual examples of each posture (see figure 4.6). An examination of these clusters reveals the similarities that exist between each posture. A range of distinct characteristics has been revealed using this method. Four relationships have been identified as being relevant to this study. First, a clusters distinctiveness can be ascertained. Second, similarities between clusters can be seen. Third, disparities in the overall consistency of replicated postures is identifiable. Fourth, the overall uniformity of replicated posture is also highlighted.

#### 4.2.0.1 Distinctiveness of hand posture

The most distinctive postures can be seen in regions of the scatter graph where there is the least density of clusters. The clusters that occupy peripheral regions of the scatter graph identify the shapes with characteristics that set them apart from other postures. The postures that are the most distinctive have been documented in figure 4.7.

#### 4.2.0.2 Similarities between hand postures

The degree of similarity that exists between postures is defined according to how much the clusters in the scatter graphs intersect and overlap (see figure 4.6 and 4.8). The coordinates of postures with similar characteristics lay in closer proximity than those that are dissimilar. For example, in figure 4.6 the two clusters representing alternative examples of the Japanese manual letter U (J_U) can be seen to overlap and inhabit similar regions of the scatter graph. The Korean manual letter N (K_N) does not overlap with J_U and inhabits a separate graph region. Through applying this method J_U has been demonstrated to show no similarities to K_N. Additionally, the naked eye shows that the American Manual letters A and S are almost identical, the scatter graph confirms that a clear resemblance exists between these postures.

#### 4.2.0.3 Uniformity of hand posture replication

Some postures can be seen to form much more compact coherent clusters than others. Coherent clusters are produced when postures are replicated in a uniform way. The postures with the most coherent and compact clusters have been illustrated in figure 4.7. Hand shape uniformity can be seen to diminish when clusters become less coherent and more diffuse. The results show that some postures have been demonstrated to contain a wide range of shape variation when replicated by different individuals. The postures that form coherent clusters represent gestures with lower degrees of shape variation. These have been identified in this investigation as the postures most likely to be consistently replicated by gesture-interface users. Postures that can be replicated clearly and unambiguously by a broad range of people are the most likely to facilitate an efficient form of interaction.

#### 4.2.0.4 Disparate results

Postures that have neither a distinct shape nor have been replicated uniformly produce disparate and diffuse clusters. Given that the uniformity of replicated postures can be identified through examining the scatter graph it is understandable that replication errors can also be identified. For instance, point Ld is in-

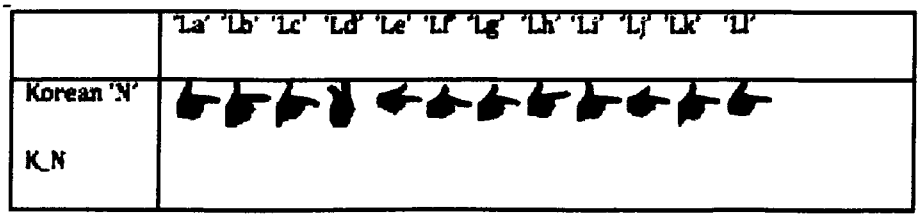



Figure 4.5: Korean manual letter N with scatter graph key

consistent with all other members of the K_N cluster (see figure 4.6). The reason for this inconsistency can be seen by visually examining the postures making up that set (see figure 4.5). Though some of the postures sampled deviate from the standard postural form they have been considered in this analysis. This creates a broader indicator for understanding similarities between gestures. Instead of focussing on the similarities of idealised postural forms the interrelation of atypical forms has also been considered. Ignoring the impact of atypical gestural form will produce a type of interaction that is incapable of predicting when errors might occur during interaction. This approach produces a representative assessment of each hand posture by considering the diversity in human hand dexterity. The postures that demonstrate large degrees of shape variance when replicated by multiple users show little or no uniformity. These posture are ambiguous and show a high probability that either misrecognition or error may occur when they are used.

### 4.2.1 Discussion

Of the fifty-two postures the participants were asked to replicate ten were represented as left-handed postures in photographic images. The majority of participants replicated all gestures with their dominant hand irrespective of whether the postures were portrayed as left or right handed. Two participants demonstrated ambidexterity when replicating postures. Notably, these participants stated they suffered from minor physical conditions that restricted their hand movements. Despite demonstrating ambidexterity neither participant demonstrated any level of sensitivity regarding which hand a posture had been portrayed to them. Though collectively participants replicated nineteen percent of

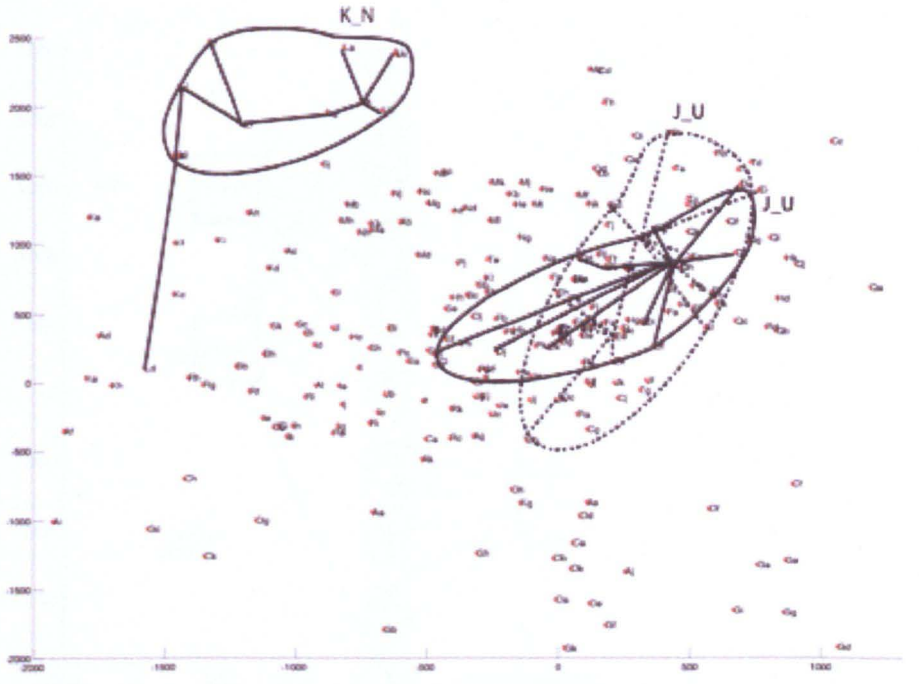



Figure 4.6: Scatter graph showing shape variation between replicated postures

all postures inaccurately, these were consistently the same errors. The errors participants made were generally as a result of the participant using the wrong hand. A small percentage of all errors made were as a result of participants replicating gestures outside of the target area, beyond the cameras viewpoint. This investigation has identified postures that could sustainably be used in computer interaction (see figure 4.3). Taken from American, Japanese, Korean and Polish manual alphabets these postures represent simple hand shapes that the average person may have used on numerous occasions. Notably, of all those tested only two postures selected by test participants required either the ring or little fingers to be extended. In both case these fingers were extended in combination with the middle finger. Consequently, less strain was placed on the users hand. Schieber [1991] shows that the middle, index and ring fingers share tendons and nerves, which make the interrelation between these fingers very high. The result of this high level of interrelation makes it both difficult and uncomfortable for each finger to move independently. The findings of this study correspond to the

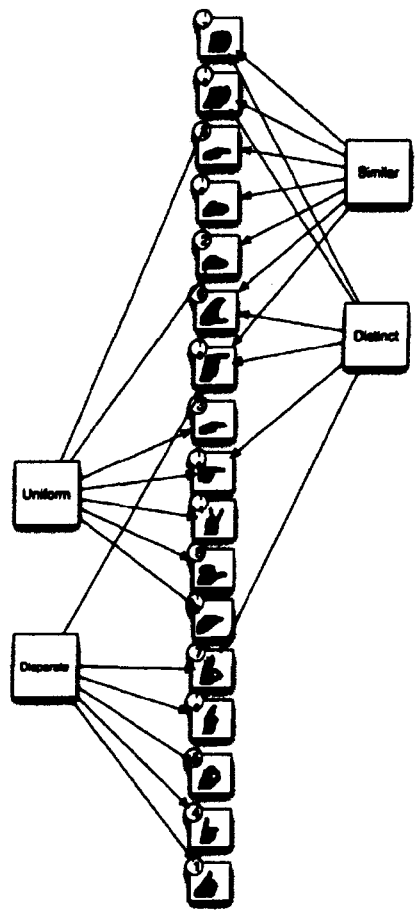



Figure 4.7: Interrelationships of postures as illustrated by the clusters in Appendix .2

conclusions of Schiebers research. The positively assessed postures were either the ring or little fingers are extended also represent common gestural expressions. The Polish manual letter o is also a symbol used to express that everything is a ok. The second posture depicts the Spanish manual letter b also communicates a desire to shake hands. The test participants familiarity with these postures might have been influential factors in the favourable assessments regarding the physical exertion experienced. In addition, the fact that these postures may have wide use and circulation suggests that the wider population has demonstrated them to be sustainable. For example, a word that is difficult to pronounce is unlikely to find frequent usage in daily communication.

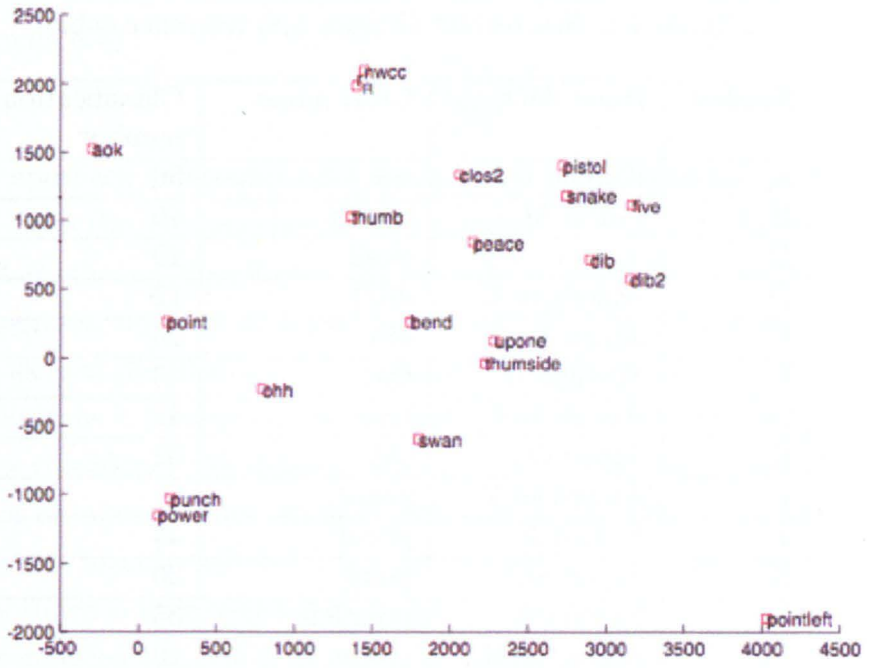



Figure 4.8: Shows the similarities between each GEf hand posture

Though collectively participants replicated nineteen percent of all postures inaccurately, participants consistently made the same errors. Due to these registration errors the array of postures comprising each set calculated in the psv index has been reduced to twelve, so that all postures have the same statistical weighting. The fact that test participant seemed to be insensitive to which hand a posture should be represented suggests that there are issues about the interpretation of posture orientation. This is an issue that has consequences on whether gestural syntax should be independent of handedness. This cannot be confirmed without further testing. Future tests would have to explicitly examine user awareness to the orientation of a posture. The similarities in the assessment of each Japanese letter u, for both the ppe and psv index, demonstrate the consistency and reliability of the overall study. Test participants were not told about the duplicate postures. Furthermore, the two postures were not proximal in the image order when portrayed to test participants. The similarity between

Table 4.1: Scatterplot (Figure 4.8) reference table

| Symbol | Hand dact | GiMI name | Classification number |
|---|---|---|---|
| A | Korean O | aok | 36 |
| B | Polish M | bendh | 46 |
| C | Japanes TO | clos2 | 29 |
| D | American U | dib2 | 12 |
| E | Japan U | dib | 25 |
| F | Spanish B | five | 43 |
| G | American C | nwcc | 42 |
| H | Japan O | ohh | 32 |
| I | American V | peace | 7 |
| J | French N | pistol | 44 |
| K | Japan SO | point | 30 |
| L | Korean N | pointleft | 35 |
| M | American S | power | 19 |
| N | American A | punch | 1 |
| O | Japan KO | R | 31 |
| P | Japan KU | snake | 24 |
| Q | Polish E | swan | 47 |
| R | Japanese TA | thumbs | 51 |
| S | Japanese A | thumside | 17 |
| T | American D | upone | 9 |

each postures evaluation can be seen in figure 4.4 and 4.6. Though these postures were replicated separately there are still shown to have the most similarities. The amount of shape variation and cluster uniformity is shown to be almost identical. If these two postures were shown to exhibit significant variation then the veracity of the results produced in this study would be in question. However, this has been demonstrated to the contrary.

### 4.2.2 A posteriori conclusions to PPE results

To check the veracity of conclusions drawn from the user test a Students T test has been conducted. Using the Matlab T test function the mean and standard

deviation of all participants responses where calculated to find the t distribution (see formula 4.4).

$$t = \frac{7.5789 - 8}{\frac{1.8048}{\sqrt{19}}} \tag{4.4}$$

This approach is recommended for use for a sample size below thirty, because in these cases the data is generally not normally distributed. An example of how user test results are distributed can be seen in appendix .1 page 159. Though the histograms depicted in appendix .1 (page 159) is close to normal they represent a skewed distribution. By applying the formula shown (4.4) a probability value known as a p-value can be obtained. In hypothesis testing the null thesis is always considered the default position. A p-value represents the statistical chance of obtaining a test statistic extreme to the default assumed. The null hypothesis is usually rejected when the p-value is less than 0.05 or 0.01. If the null hypothesis is ever rejected the result is considered significant. In the case of this investigation the null hypothesis is framed as follows. The null hypothesis in question assumes that using a questionnaire for classifying physical exertion within a sample group will fail to find relationships within sample data, unless purely by chance (See formula 4.5). The alternative being that questionnaires can be effective methods for producing generalised models of perceived user exertion. The method employed is robust enough to facilitate the creation of a coherent gesture comfort index (See formula 4.6).

A null response from this test means that the study cannot reject the null hypothesis. However, alternative responses will suggest that potentially significant conclusions can be drawn from the data. Figure 4.9 illustrates the conclusions of the T test. The image represents a two-dimensional 19 by 52 matrix. Each row along the horizontal axis represents a different participant of the user study. Each column along the vertical axis represents hand postures as they are listed in figure 4.1. If the pixels in figure 4.9 are black this represents a null response to the t test. However when examining the pixels in figure 4.9 no black pixels were found. The conclusions of the T test suggest that it is safe to reject the null hypothesis. Enabling the study to state with confidence that the user questionnaire applied in this study has allowed a generalised model of perceived user exertion to be calculated.

$$H0 : S \leq 7 \vee \geq 2 \tag{4.5}$$

$$H1 : S > 7 \ \& \ \sigma < 2 \tag{4.6}$$

#### 4.2.2.1 Additional support for priori findings

When applying a similar calculation to compare the mean PPE score of each gesture further evidence to support the method of classification defined in section 4.1.5 (page 79) have been found. The calculation used the same null hypothesis as the default position expressed by the expression 4.5.

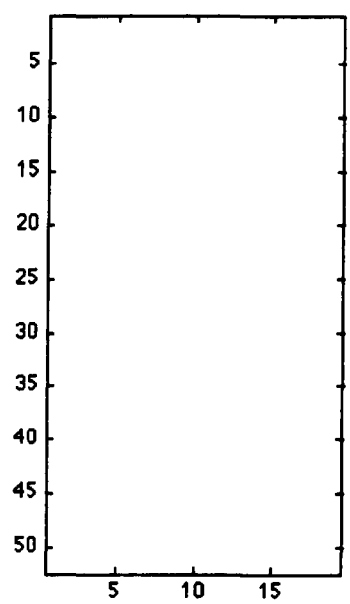



Figure 4.9: Student T test examining mean responses for each gesture

Figure 4.11 illustrates how classifications defined using a priori set of benchmarks have been supported through the use of the T test. The image represents a two-dimensional 52 by 52 matrix. Each row along the horizontal axis represents a different hand postures (see figure 4.11). Each column along the vertical axis also

represents hand postures as they are listed sequence in figure 4.1. Once again If the pixels in figure 4.11 are black this represents a null response to the t test and nothing definitive can be concluded from the PPE scores. Examining figure 4.11 it is clear to see that there is variation in the T test results. However upon further inspection it becomes clear that the rows and column that contain predominantly white pixels correlate directly with the postures defined as neo-optimal (see figure 4.3). To examine the correlation between results examine figure 4.3. The circles in the top left corner of each silhouetted posture contain a number that is paired to the numbers represented on the x,y axis of the matrix illustrated in figure 4.11.

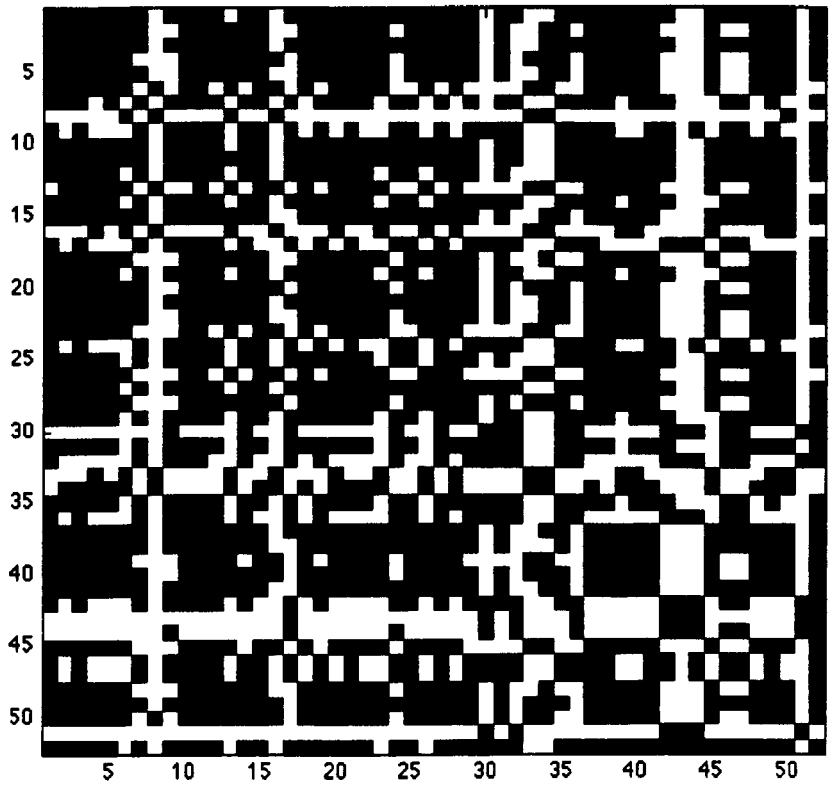



Figure 4.10: Using T test to compare the each gesture PPE score significants (xy axis represent hand posture see figure 4.1)

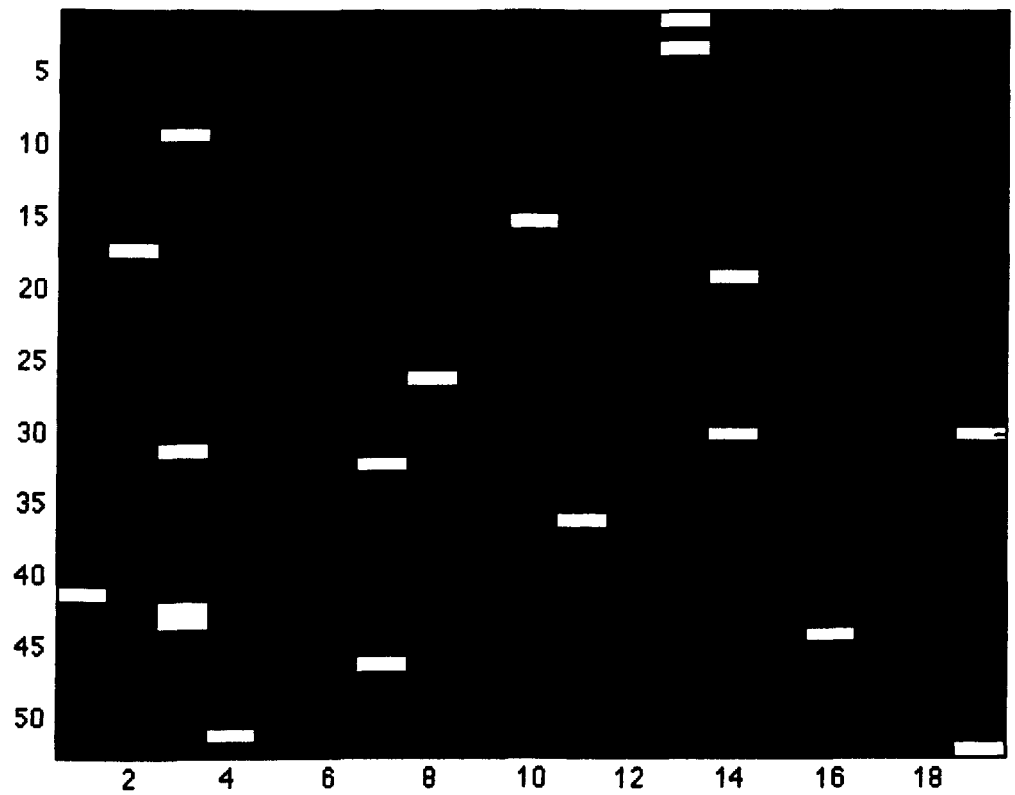



Figure 4.11: Using a T test to identify extremes beyond the baseline PPE mean (X = users, Y = GEf postures)

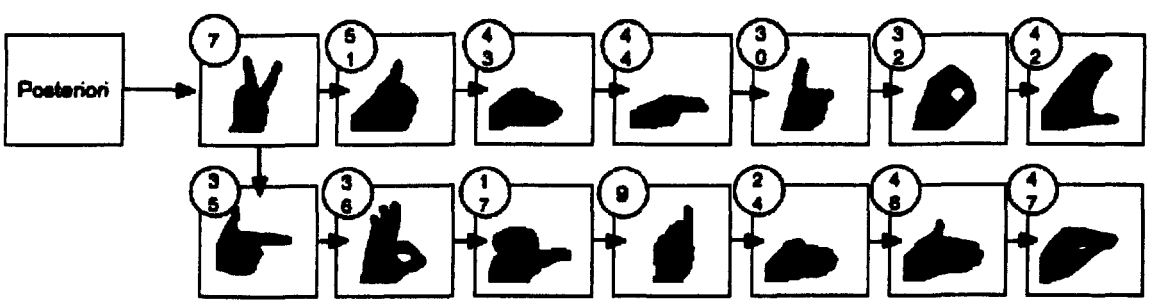



Figure 4.12: Gestures with PPE scores identified as significant in a T test

### 4.2.3 A posteriori conclusions to PSV results

When examining the potential shape variation of user responses two different statistical analysis methods were utilised. Principal component analysis was used for the purpose of allowing postures in the GEf dataset to be visually compared on a single scatterplot. This enabled this study to produced a method for looking at similarity between postures (see appendix .2 (see figure 4). Cluster sizes, proximities and overlapping were identified by hand. Additional to this method the Mahalanobis distance was calculated and express as a percentage (see appendix .2 figure 3). The table compiled includes the mahalanobis distance results in addition to a PCA cluster size reference. The closer the mahalanobis distance is to one-hundred the greater amount of similarity has been calculated. These distances are easier to see when the amount of principal components are reduced (see 4.8 and table 9).

### 4.2.4 Unresolved issues

Despite the successes this research has not implemented a method for evaluating the memorability or the intuitiveness of a gesture lexicon. These are issues which will require further investigation at a later stage. This should not however diminish the fact that through this research a set of hand postures can confidently be defined as adhering to generalised comfort thresholds. However, the methods and framework outline in chapter 2 (see page 50) seem to have been justified by the conclusions of this user study. Figure 4.13 illustrates the current relationship between the principles discussed in chapter 2 and the approach to classification presented in this chapter.

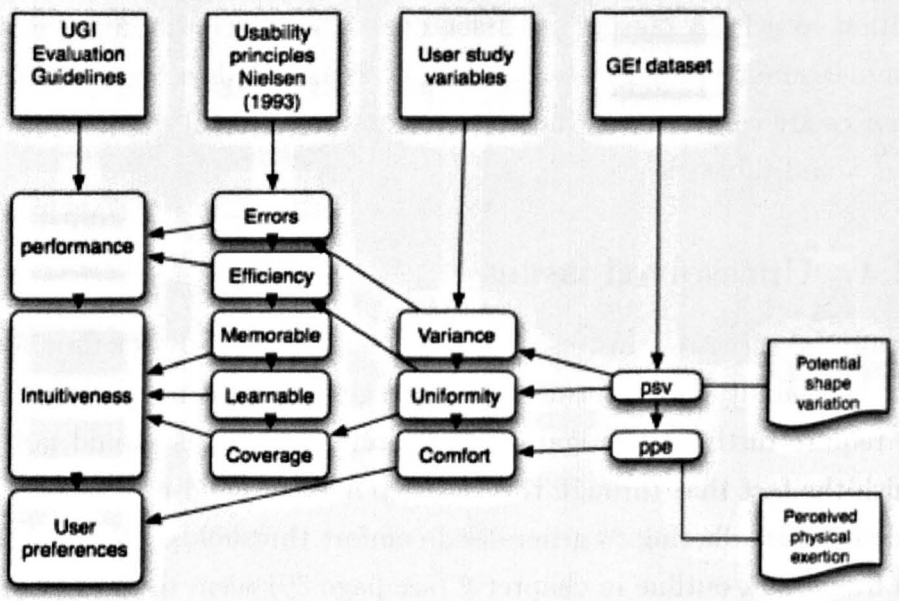



Figure 4.13: An iterative contextualisation of research guidelines

# Chapter 5

# GiMI

Chapter 5.1 presents the GiMI syntax model devised during this research. The elements and structure of this syntax are described, and in addition the benefits this syntax model offers are contrasted against other models.

## 5.1 Gestures in Machine Interaction Syntax Model (GiMI)

Once the capabilities of people have been prioritised, the need to separate a systems functional apparatus from a user's physical interactions becomes increasingly evident. At some stage in the evolution of unencumbered interface development a formal distinction between the action required to mediate an interaction and the mechanical system confronting the user will be needed. At such a stage it might be sensible to refer to the physical mechanism as the interface and the actions required to operate the system as the syntax. This will become increasingly the case when interfaces become receptive to increasingly complex modes of input and interaction. Emphasising these issues during interface development may limit the cycle of obsolescence to apply only to hardware not UGI syntax. For the purposes of clarity in this chapter when discussing the action required to mediate an interaction the term syntax will be used. A gesture syntax model, designed to enable robust and efficient interaction with gesture-face layers (GFL), is presented in this chapter. The syntax has been designed to facil-

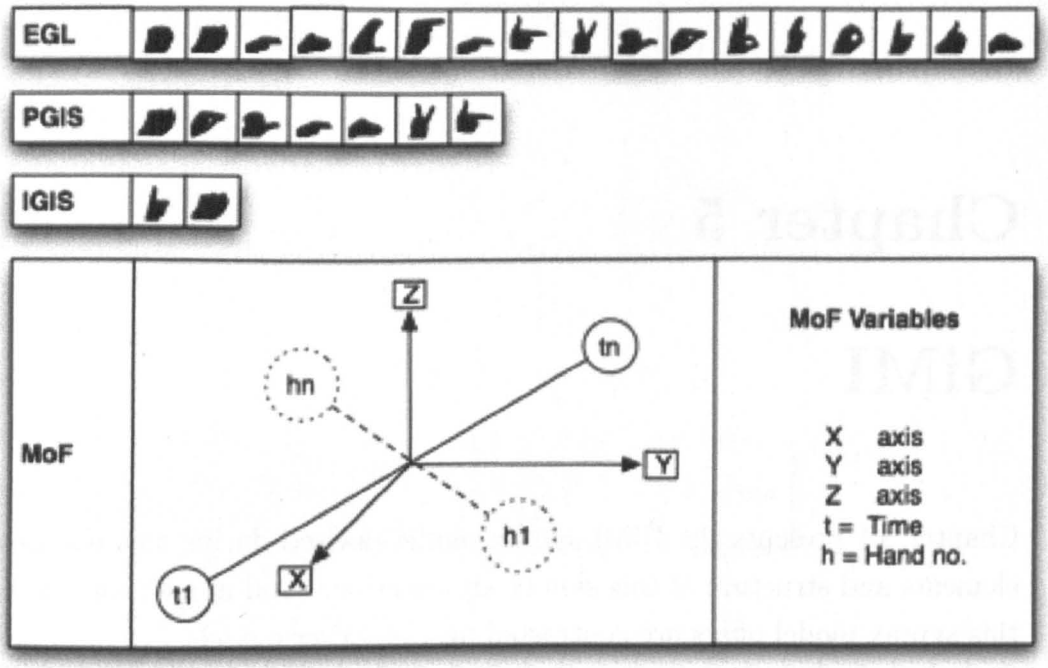



Figure 5.1: Illustrates the structure of GiMI syntax

itate ergonomic interaction with gesture-recognition systems. This investigation presents an ergonomic framework for using deictic ergotic and semiotic gestures during computer interaction. Utilising these gestural impulses will enable robust and versatile syntax to be developed.

### 5.1.1 Extended Gesture Lexicon (EGL)

The first class of gestures defined within the GiMI model is the extended gesture lexicon (EGL). The postures included in the EGL have been identified as comfortable postures and fulfil the requirements of both the neo and beta comfort thresholds defined in chapter 4 page 74. The postures identified by the neo and beta thresholds represent postures that have been evaluated to be physically sustainable and ergonomic (See figure 4.3 page 79). These postures are documented within the ppe index of the Gesture efficiency (GEf) dataset (see chapter 4 and Appendix .1). Currently there are twenty postures that have been identified as offering optimal comfort. However, as the GEf dataset is an iterative framework

there is potential for the EGL to increase in size, as new postures are evaluated and added.

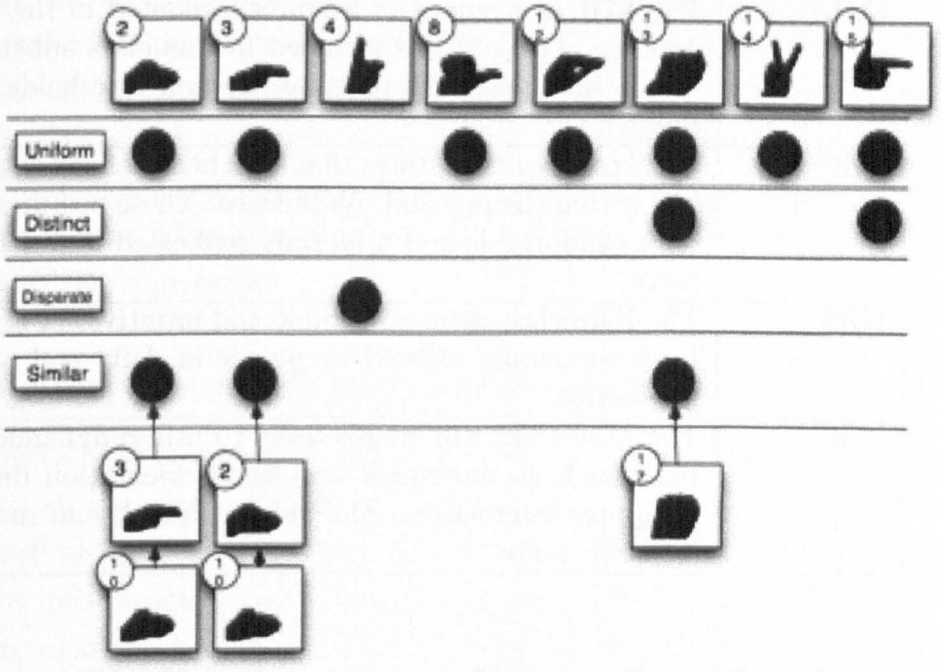



Figure 5.2: Illustrates how PGIS postures perform within the psv index

### 5.1.2 Performable Gesture Instruction Set (PGIS)

The second class of postures defined within the GiMI model is the Performable Gesture Instruction Set. These postures are a subset of the EGL and currently consist of seven postures. These postures adhere to both optimal ppe thresholds and those defined as either uniform or distinct, within the potential-shape-variation (psv) index. These postures are illustrated in figure 5.1 and 5.2. These postures have been evaluated in the GEf dataset to be the most likely to offer efficient and ergonomic interaction.

Table 5.1: describes the four classes constructing the GiMI syntax model

| GiMI class | Description |
|---|---|
| EGL | The EGL is a subset of gestures evaluated in the GEf dataset. The gestures included in this class adhere to the optimal perceived physical exertion thresholds (see chapter 4). |
| PGIS | PGIS represents postures that have been defined as optimal within the ppe and psv indexes. These postures are both comfortable and uniformly replicated by interface users. |
| IGIS | The IGIS class defines a simple and intuitive set of gestures commonly utilised by people in daily real world interaction. |
| MoF | The MoF class will enable users to utilise dynamic actions such as movement and hand orientation during computer interaction. MoF utilises depth and motion disparity maps. |

### 5.1.3 Intuitive Gesture Instruction Set (IGIS)

The third class of the GiMI model is the Intuitive Gesture Instruction Set (IGIS). The criterion for a postures inclusion into the IGIS is that the posture be intuitive and comfortable to perform. These gestures represent natural gestures that either have a predefined meaning or function. IGIS posture represents postures that should adhere to optimal ppe index thresholds, though may demonstrate significant variation within the psv index. The primary purpose of the IGIS is to facilitate intuitive interaction with computers. The postures currently included within the IGIS are the deictic pointing and ergotic steering gestures (see figure 5.1 and 5.2). In figure 5.2 these posture are labelled 4 and 12, respectively. These postures have specifically been selected because they are both universal and intuitive. In addition to being intuitive they also have been evaluated to be physically sustainable (see Appendix .2). These postures have been included in this syntax model even though the deictic pointing gesture showed significant capacity for shape variation (see psv appendix .2), specifically when using standard 2D image templates. As both of these postures project forward beyond the 2D plane, cre-

ating accurate 2D image templates is difficult. Nevertheless, by utilising depth silhouettes this research identifies a robust and efficient method for accurately recognising these postures (see chapter 3 page 53). The primary function of the IGIS model is to provide a simple set of hand signs that enable users to intuitively navigate an interface. To facilitate rapid and intuitive user interaction the lexicon of each IGIS class is limited to a small set of signs that work in conjunction with simple directional hand orientation, such as up, down, left, right and twisting motions. By including these gestures into the GiMI syntax framework a versatile model of UGI is presented.

### 5.1.4 Gesture modifying function (MoF)

The fourth class within the GiMI syntax model is a modifying function (MoF). In isolation the postures comprising the EGL represent static semiotic posture. The MoF provides the capability to extend the meaning of individual EGL postures by incorporating a set of parallel actions such as the movement, orientation the interrelation of hand postures. The context and meaning of EGL postures are modified by the accompanying MoF function. The addition of this function enables the syntax to grow in complexity as the proficiencies of users develop.

### 5.1.5 Gestures in machine interaction (GiMI)

Through the combination of these classes robust interaction can be implemented on computers. These interactions include the capability to engage in cursor control drag and drop together with steering activities. The postures comprising each GiMI function can be seen in figure 5.3 (page 103) and 5.5 (page 109).
GiMI functions:

- GiMI_pl: Gesture-Face Layer
- GiMI_p_c: Point & Click
- GiMI_d_d: Drag & Drop
- GiMI _s_hi: Select & SelectALL

- GiMI_del: Draw & Erase
- GiMI_fwd_rev Forward —— Reverse

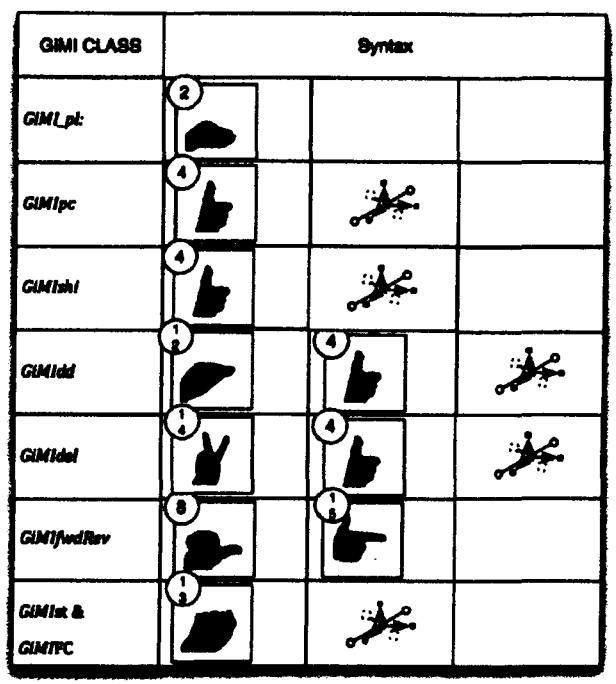



Figure 5.3: Illustrate the classes utilised in each GiMI function

#### 5.1.5.1 GiMI, place the Gesture-Face Layer class (GiMIpl):

The GiMI_pl class defines the location of the Gesture-Face Layer (GFL). Through the use of stereo disparity mapping algorithms depth metric information is obtained. The resulting metric information has been used to define virtual spaces and surfaces. The surface created can be programmed to be sensitive to the proximity of objects and people. The GFL has been defined using this method, enabling touch screen like interaction that is virtual opposed to physical. The GiMI_pl function utilises the Spanish manual letter M. This posture has been defined as one of the most comfortable gestures to replicate. Within the psv index this posture also demonstrates a high level of shape uniformity, which suggests that a wide range of users will be able to replicate it consistently. The Spanish

B posture is a command used for defining the position of a Gesture-Face Layer (GFL). Wherever the Spanish B is replicated the computer identifies its location in space, using depth metric information. The proximity of the hand is calculated and the GFL is mapped to these specific coordinates. Once the GFL interface has been placed using this action other postures can be used to interact with the GFL.

#### 5.1.5.2 GiMI point and click class (GiMI pc):

The point and click class (GiMI pc) is designed to facilitate intuitive and accurate GUI interaction. Using depth information the GFL is able to track the proximity of the hand in relation to it surface. The posture utilised within this function is one of the most natural and intuitive available. Once the GFL has been placed the deictic pointing gesture can be used to interact with the virtual surface created. By tracking the location of this posture the computer is able to detect when the finger breaches the surface of the GFL. Breach points identify instances of interaction in the same way that a mouse click symbolises a selection request. To ease the cognitive load placed upon those applying this syntax augmented visual feedback is presented to the interface user. In this example a cursor is used. In addition, to facilitate unambiguous interaction an outer-tracking-layer (OTL) is placed between the user and the GFL. The secondary layer sits five to ten centimetres beyond the GFL and is also defined using distance metric analysis. The tracking layer enables the extended finger to be tracked prior to the breach of the GFL.

#### 5.1.5.3 GiMI drag and drop class (GiMI dd):

The GiMI dd is designed to facilitate the physical relocation of virtual objects, such as desktop icons. The GiMI dd extends the functionality of the GiMI pc with the addition of the Polish manual letter E, which is one of the most distinctive and consistently replicated postures. The Polish E is a posture that is distinct and accurately recognisable through 2D image templates. The form and shape of this posture corresponds to the pinch action with the thumb and middle finger meeting at their tips. The pinch is ergotic and enables people to pickup or squeeze objects.

In the GiMI dd function, the Polish E posture provides a symbolic representation of the ergotic pinch action. The replication of this posture instructs the computer to reinterpret subsequent GFL breaches. The functionality of the GFL changed as a consequence. Rather than just selecting regions of interest, as with the GiMI pc, physical contact with the GFL will facilitate the relocation of virtual objects. When in GiMI dd mode, the first breach identifies the location of the object of interest. The second GFL breach instructs where the object should be relocated. Once the object has been moved the GFL automatically switches back to GiMI pc mode.

#### 5.1.5.4 GiMI delete class (GiMI del):

The GiMI_del is a gestural function designed to facilitate the deletion of computer data. The GiMI_del extends the functionality of the GiMI_pc with the addition of the American manual letter K, which is also one of the most distinctive and uniformly replicated postures. The American manual letter K is distinct, consistently reproducible and accurately recognised using 2D image templates. The American K is used as a symbolic representation of a pair of scissors. When a virtual object has been selected, using either the GiMI pc or GiMI shi class, mimicking the scissor action with the American K posture will initiate GiMI Sdel and delete the object.

#### 5.1.5.5 GiMI steering and control class (GiMI_Mst):

The GiMI steering function is a gesture syntax model designed specifically for the control and navigation of robot agents. Robot agents are semi autonomous devices such as remote control robots or vehicles. By extracting metric information from depth silhouettes the positions of GiMI_Mst postures are tracked. The GiMI_Mst function behaves like a steering wheel except that it has no physical steering column. To steer using GiMI_Mst function both hands must replicate the posture representing the American manual letter A.

When both hands are positioned in parallel along a horizontal axis the GiMI_Mst function instructs the robot agent to travel in a straight line. When the left hand is pull down below the right the robot agent is instructed to turn left. Inversely,

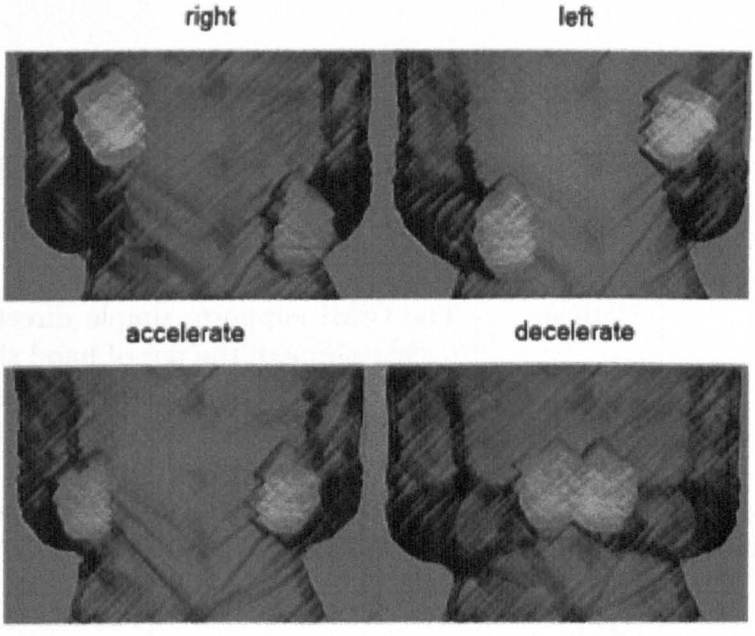



Figure 5.4: Shows gestures used in GiMI steering model

when the right is pull down below the left the agent is instructed right. The vertical distance that separate each hand corresponds to turning angle instructed in the agent.

#### 5.1.5.6 GiMI Speed and Velocity class (GiMI_FC):

The GiMI velocity function is similar to the GiMI_Mst function in that it has been designed for the control a robot agents. The GiMI_FC function also relies on depth metric information. To control the velocity of a robot agent both hands need to replicate the American A. The velocity of a robot agent corresponds to the distance that separates both postures, which is measured along the horizontal line. The closer the hands the slower the agent is instructed to travel. The further apart the hands the faster the agent is instructed to go.

Table 5.2: GiMI function using either the IGIS and PGIS lexicon

| Classification / Taxonomy | Context / Task | Prototype description |
|---|---|---|
| GiMI | Main set | The GiMI supports simple direct navigation of an interface through the use of hand shapes and orientation, such as up and down. . Replacing some of the functionality of a mouse or pen stylus with a hand tracking and shape detection algorithms. |
| GiMI_pl Symbolic | Placing Gesture-face-layer (GFL) | To position and location of the Gesture-face layer is defined using the GiMI class. |
| GiMI_p_c Deictic | Point and Click | The position of a pointing gesture is mapped to the location of an on-screen cursor. An outer tracking layer tracks the hand when it is not in contact with the Gesture-Face-Layer. |
| GiMI_shi Deictic and Beat | Select and Highlight | To select and highlight a passage of text from a document, a variation of the GiMI_p_c can be used. Regions of interest can be communicated by moving the finger whilst maintaining contact with the Gesture-face layer. This creates a selection box. |
| GiMI_d_d Deictic and Symbolic | Drag and Drop | Using the middle finger and thumb to pinch a virtual object will initiate the drag function. An object can then be moved and by moving the middle finger and thumb apart and pointing with the index finger, can be dropped into position. |
| GiMI_del Deictic and Symbolic | Delete / Cut | This function can be used to delete a passage of virtual objects or text. Once the object of interest has been selected by the pointing action it can be deleted, using a combination of GiMI_pc and the American manual letter K posture |

Table 5.3: GiMI function using MoF

| Classification / Taxonomy | (MoF) Hand | Reference | Context Task | Description |
|---|---|---|---|---|
| GiMI_Mst | Subordinate hand Dominant hand | American A American A . | Turn right / left | Depending on which hand is in the higher position determines the direction of the GiMI_Mst function. |
| GiMI_FC | Subordinate hand Dominant hand | American A American A . | Speed control | The proximity of each hand determines the speed of the robot agent. The closer the hands are the slow the agent. The greater the distance between each hand the faster the agent. |

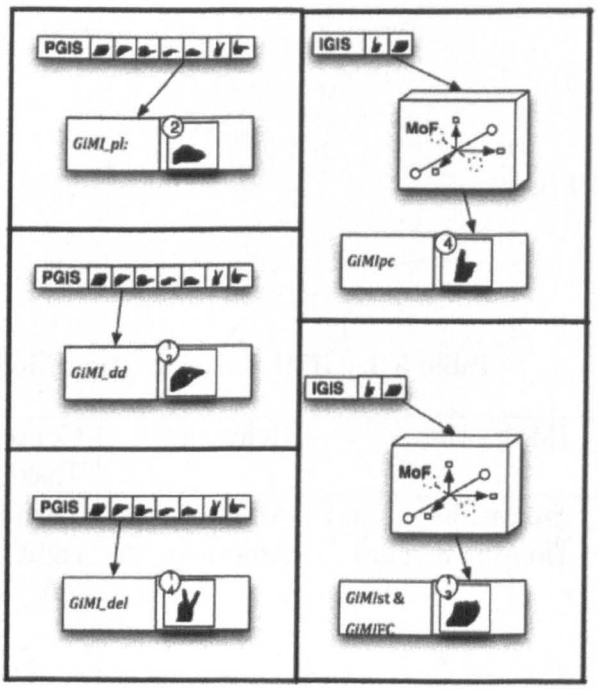



Figure 5.5: Illustrate how GiMI classes interact to provide unencumbered interaction

## 5.2 Discussion

The motivation underlying the design and development of the GiMI lexicon is to demonstrate that through utilising a development framework that prioritises user preferences in the early stages of interface development an ergonomic gesture lexicon can be created. A range of methods used by usability researchers and engineers have been explored and a framework for evaluating the ergonomics of UGI has been established in tandem with the development of an interface prototype. An iterative framework has been developed so that the performance of users can be evaluated in parallel to the recognition accuracy of gesture lexicons. By understanding these issues the lexicon created attempts to demonstrate that appropriate uses of gesture which maximise our performance with machines can be constructed.

# Chapter 6

# Conclusion

Chapter 6.1 summarises the aims and underlying motivations driving this research inquiry. Chapter 6.2 summarises the journey taken during this research; describing the underlying motivations driving this investigation and detailing the challenges encountered. This section provides a context to research outcomes that are presented later in the chapter. Chapter 6.3 defines the outcomes of this research; presenting study and design contributions in the field of unencumbered gesture interaction. Chapter 6.4 discusses the contemporary arena of interface development where this research takes place, discussing the potential of unencumbered gesture interaction in relation to other interface models. Chapter 6.5 presents a critical appraisal of this investigation and discusses grounds for further research.

## 6.1 Summarising research arguments

This research sought to investigate whether an ergonomic model of UGI can be developed and implemented on consumer devices. It also investigated the types of barriers preventing UGI from being widely adopted. This research aimed to engage in the development of freehand gesture interfaces and accompanying syntax and provide a roadmap for developers of the field, so the development of un-ergonomic, inefficient interfaces could be avoided. There were two underlying motivations underpinning this research investigation. The first motivation was to define a straight forward method for developing and testing gesture lexicons in

parallel. The second motivation was to design and implement a computer-vision interface capable of recognising a diverse form of gestural actions. Actions that include the deitic, ergotic, epistemicand semiotic forms of gesture.

## 6.2 Summary of outcomes

In pursuing the development of an ergonomic gesture interface, this research develops methods for designing and optimising the performance of interface syntax. To achieve these ends this investigation has examined the fields of computer-vision, gesture research and ergonomics. Through the process of implementing optical gesture recognition and creating efficient image-processing frameworks (Chapter 3, page 53 ), this research has encountered many obstacles, overcoming these with the discovery of robust solutions. These have included successfully constructing 4D image templates through the use of motion and depth disparity maps and training computers to recognise gestures by using statistical algorithms such as haar-like feature detection, PCA and mahalanbois distancing. The system outlined enabled this investigation to produce an interface capable of accurately distinguishing gestures in changing environments and under variable light conditions (Chapter 5, page 98). Through the practical implementation of these techniques, an optimal system configuration for recognising gesture is presented and advocated. Through solving these challenges a robust syntax model, which incorporates wide range of gestural actions has been developed. As this research focuses on developing an ergonomic model of freehand interaction, methods for evaluating interface performance and user preference was needed. In investigating these methods, guidelines for iteratively evaluating the performance of gesture interface syntax were produced (Chapter 2, page 35). The methods produced have been practically applied in a series of studies that evaluates the physical exertion of people (Chapter 4, page 74). By identifying the performance constraints of people and computers, critical elements underpinning future interaction have been defined. This investigation provides a framework that shows gesture interface developers how to design gesture syntax for unencumbered interfaces. Dynamic and ergonomic syntax that facilitates interaction with mixed-reality en-

vironments, robots and vehicles can be produced as a result of implementing these frameworks.

## 6.3 Outcomes of research

Through pursuing this research a range of contributions have been made. Offering empirical, theoretical, design and methodological contributions to the fields of interface design and ergonomics. The outcomes of this investigation contribute to two fields of research. First, the physical ergonomics of gesture interaction has been advanced through the development of a syntax evaluation framework, enabling greater understanding of an emerging field. Second, contributions to the field of interface design have been made through the development of a gesture syntax model and design methodology. Consequently, a robust and effective optical recognition system has been defined, freeing future developers to concentrate solely on the creation of novel applications, instead of focussing on how best to configure the physical interface. Understanding the physical ergonomics of target users is conditional to being able to recognise the limitations of human actions. Therefore, assessing the physical performance and preferences of likely users was also essential to this research. However, to determine the performance of optically mediated interaction the computers capability to recognise gestures also had to be measured. Conducting these investigations in tandem allows the effectiveness of syntax to be reliably predicted. Methods for evaluating the performance of gesture syntax have been produced as a result of this process (Chapter 4, page 74). The outcome of this process will help ergonomic evaluators with limited knowledge of computer-vision and gesture research to evaluate gesture syntax. Additionally, this research will help gesture interface developers create ergonomic interfaces for users.

### 6.3.1 Practical study: implementing and testing a gesture interface

Specific contributions that result from the outcomes derived from this investigation are discussed in this section. Consisting of four theoretical, one methodological and an empirical set of contributions. The section is structured as follows.

- Theoretical
  - Comprehensive review of the field of UGI
  - Criteria to determine the success of an interface
  - Mapping the taxonomies underpinning UGI research
  - Defining physical and cognitive constraints of gesture
- Empirical
  - Definition of user preferences and gesture efficiencies
- Method
  - Presenting guidelines for ergonomic UGI development
  - Derived a method for creating ergonomic UGI frameworks

#### 6.3.1.1 Comprehensive review of the field of UGI

A detailed review of the field of UGI from its inception to the present date has been undertaken in Chapter 1 (page 1 - 29). This review provides readers with a historical account of pioneers within the field of UGI and offers an insight into the type of technologies and techniques that have been utilised in the development in unencumbered gesture interfaces. Readers of this review will be able to consider both the successes and failures of previous developments, enabling them to avoid repeating unnecessary steps. In addition to these outcomes, this review and knowledge of prior research has aided this investigation in defining specific criteria that can determine the likely success of an interface.

#### 6.3.1.2 Criteria to determine the success of an interface

Through examining the field of UGI development it became increasingly apparent that there were three clear indicators that could be used to determine the potential success of an interface. Firstly, the coherence of a physical interface was a significant factor determining its potential. A second factor helping to determine success was whether an interface provides a previously unavailable function or was an improvement on pre-existing functions. Thirdly, ease of use and the versatility of the underlying interface syntax is a factor in determining its underlying promise. Though individually each criterion may seem obvious, in order for an interface to be widely accepted by users, all three criteria had to be evident. Examples of where an interface has failed to fulfil each requirement have been discussed in Chapter 1 (pages 29 - 34). All of the criteria described when combined satisfy key goals of usability, which is to provide good utility. A coherent physical interface coupled with efficient syntax will likely facilitate interaction that is both intuitive and efficient. By following these criteria developers will create interfaces that are both easy to use and learn, enabling users to perform efficiently.

#### 6.3.1.3 Physiological and cognitive constraints

In order for the field of UGI to develop ergonomically it is important that developers recognise the limitations of human action. After identifying the apparatus used during gestural activity this investigation has been able to define inherent constraints of gesture. For example, this research contrasts the performance of writing, speaking and signed language and further demonstrates that gesture could speed up and enhance the way we interact with computers. However, this investigation also highlights the risk associated with relying solely on semiotic sign language (Chapter 2, pages 41 - 44). Demonstrating the benefits and risks associated with gesture interaction this research intends to steer developers along a path of ergonomic interface and syntax development. Through recognising the concerns uncovered in this research future UGI developers will realise the importance of creating syntax that incorporates a broadest range of gestural impulses. Such syntax may allow users to engage in a more intuitive mode of interaction, as the syntax would be more representative of natural gesture. Though this research

has not determined the merits of using natural gesture as opposed to artificial syntax, it has demonstrated that gesture interaction is less effective when particular facets of gesture are used in isolation (Chapter 3, page 62 - 56).

#### 6.3.1.4 Defining user preferences and gesture efficiencies

A dataset is compiled to catalogue the physical preferences of gesture interface users. This is conducted in parallel to monitoring the underlying performance of users when replicating gestures (Chapter 4, page 74 - 98). The dataset examines the needs of users, placing them central to the design process. The subsequent dataset provides benchmarks that can be used to gauge the underlying efficiencies of specific gestures. The dataset enables gestures to be assessed independently of hardware limitations. For example, gesture syntax can be developed and assessed independently of the construction of physical interfaces due to the availability of a coherent evaluation framework. Such an approach will enable developers to build and design interfaces around gesture syntax as opposed the making syntax structure conditional to the composition of hardware architecture. Empirical information detailing variations in users performance together with their physical preferences is documented in the Gesture-efficiency (GEf) dataset. All accompanying data and analysis is presented in Appendix 1.1 and 1.2. Using the dataset of gesture efficiency, together with syntax assessment guidelines (Chapter 2, pages 44 - 53), a method for optimising gesture syntax in machine interaction can be applied.

#### 6.3.1.5 Guidelines for developing an ergonomic UGI framework.

A set of guidelines that help evaluators examine UGI syntax has been developed as a consequence of the practical implementation of optical gesture recognition and research of human physiology (Chapter 2, pages 44 - 53). The guidelines have been designed to make developers of unencumbered gesture interfaces more sensitive to the preferences of users. These guidelines have been utilised in the course of this investigation through a user study, which evaluates both the performance and preferences of people (Chapter 4, pages 74 - 98). Through these guidelines a

proactive approach to inclusive design is advocated and a detailed evaluation of user preferences and capabilities is encouraged. The guidelines present an iterative method for evaluating gesture, which enables other independent evaluators to develop and revise the dataset of gesture efficiency (GEf), which has been compiled during this investigation.

#### 6.3.1.6 An ergonomic UGI development approach.

Determining the underlying ergonomics of an interface and accompanying syntax was the primary pursuit of this research. In order that this could be achieved a method for assessing the performance of an interface together with its users had to be devised. As UGI is still emerging as a field of research an archetypal interface model is yet to be established. As a consequence there is no standard method that can be utilised by usability evaluators. To determine the underlying ergonomics of freehand interaction this research conducted an investigation using the following approach. This research first sought to identify what is likely to be the structure and composition of future UGI, by conducting a review of the field. Aided by subsequent findings, which identified pioneers within the field in addition to criteria determining the success of an interface an archetypal model of UGI could be predicted. The model defined is similar to the VIDEOPLACE interface model developed by Krueger et al. [1985]. However, through the practical assessment and implementation of state-of-the-art image processing algorithms this research is able to recommend modification to the archetypal model described, thus demonstrating an optimal system configuration for UGI evaluators. By determining the likely configuration of a UGI framework this research has been able to evaluate the performance of gesture syntax (Chapter 4, pages 74 - 98). However, before a robust evaluation framework could be conducted a range of issues had to be addressed. To examine the underlying performance of gesture syntax it is essential that the criteria of the investigation be clearly defined. For example, when evaluating performance the effectiveness of a gesture can either be defined by a computer's ability to recognise the gesture or a persons ability to accurately perform the action. Researchers of this issue have tended to evaluate the abilities of computers to recognise a persons actions as opposed to evaluating a persons

ability to perform those actions. However, recently some researchers have realised that this is not an effective method of developing ergonomic interfaces (Chapter 2, page 44). Though the importance of user performance have been accepted as important, research issues the intricacies of optimising syntax performance is not researched in sufficient detail, as there are many variables to consider when evaluating the performance of gesture interface syntax. Whilst both, user performance and computer recognition accuracy need to be considered, user comfort needs to be measured. Stern et al. [2006] come close to developing a comprehensive evaluation framework. However, they admit their methods for evaluating user comfort could be improved. In this investigation, the parameters of this issue are redefined to place greater emphasis on user preference as opposed to comfort. By redefining the issue in this way this research is able to confidently identify the types of gesture that are preferred by users. Combining this knowledge with what is known about human physiological constraints this research has been able to define an optimal gesture syntax model. In order to select the gestures that can most effectively be utilised there are a variety of issues that need consideration. First, in order to use multiple gestures in a single syntax each gesture needs to be distinguished from the other, consequently each gesture needs to be distinctive. Second, if the syntax is to be performed by multiple users the lexicon of gestures must demonstrate that they can uniformly be performed by a large sample, as the higher the shape variation exhibited during the replication of each posture the lower the potential for accurate recognition. Third in order that gesture syntax can be reliably performed, the gestures utilised needs to be both comfortable and not require high levels of exertion. As previously mentioned this third category was redefined to consider the preferences of users. The combination of these issues determines the underlying performance efficiency of gesture syntax. For example, though a gesture may exhibit distinct characteristics such as in the semiotic gesture OK, where the thumb and index finger meet to create a circle leaving the remaining finger to point upwards; this does not mean that it will be uniformly replicated by a large sample. As a result the OK gesture may not an ideal candidate to incorporate into an ergonomic gesture syntax model. Though the ability to test how effectively computers can recognise gesture is very useful when developing gesture syntax, it is not vital as the outcomes of this research provide

developers and evaluators with a dataset and a development framework. Consequently gesture syntax can be investigated without compiling datasets, which is very time consuming, if evaluators use the dataset and methodology defined in this thesis. Despite the successes this research has not implemented a method for evaluating the memorability or the intuitiveness of a gesture lexicon. These are issues which will require further investigation at a later stage. This should not however diminish the fact that through this research a set of hand postures can confidently be defined as adhering to generalised comfort thresholds.

### 6.3.2 The syntax design: methods, outcomes and impact

Contributions that result from the implementation of an optical gesture recognition system are discussed in this section. Consisting of two design-based and two methodological sets of contributions, the section is structured as follows.

- Design-based
  - Gesture-face interface;
  - GIMI syntax;
- Method
  - Robust optical gesture recognition system;
  - Comparative syntax evaluation;

#### 6.3.2.1 Gesture-face-layer

An unencumbered gesture interface has been developed and outlined (Chapter 3, pages 54 - 56). The gesture-face is a digital interface capable of modelling and recognising activity within the visible environment. A system computer functions can be superimposed on to 3D real world in real-time. The Gesture-Face layer is designed to allow users access the computers through the use of gesture alone. This research advocates this model as optimal for unencumbered gesture interaction.

#### 6.3.2.2 GiMI syntax

The syntax developed in this study has been designed to facilitate versatile and sustainable interaction with computers. Through this research both the Gesture in Machine Interaction syntax model and the Gesture-Face Layer are introduced as a solution to sustainable gesture interaction. The GiMI syntax includes a model for cursor control and interaction together with a model for remote vehicle steering and control (Chapter 5, pages 98 - 108). The interface syntax created provides a framework that allows users to interact with computers using a combination of natural and artificial actions. This research presents a framework for developing optical gesture interface environments in which users need not resort to the awkward command vocabulary of keyboard-and-mouse interaction. Furthermore, the syntax created enables multiple gestural impulses like semiotic, deictic and ergotic actions to be incorporated into the architecture of graphic user interaction. The interface syntax created has been iteratively evaluated using the GEf dataset. In Chapter 5 (page 108 - 108) the GEf is used to compare the performance of competing gesture syntax models.

#### 6.3.2.3 The potential impact of contributions

By using state-of-the-art image processing and pattern recognition algorithms, such as depth, motion and shape detection, this study has been able to create 4D image representations of gesture. The use of these techniques has enabled the investigation to implement methods for utilising many different types of gesture in UGI. Consequently, this research has been able to developed syntax that utilises ergotic, semiotic and deictic gestures. Syntax that facilitates intuitive and versatile interaction has been developed in parallel with a cohesive and stable interface. Though the syntax created in this study uses artificial gestures, it maintains the link between natural motor impulses and activity type. For example, the GiMI steering model utilises ergotic actions to perform an object manipulation tasks in contrast to using semiotic actions to represent ergotic tasks. Utilising ergotic actions in this way enables the link between natural gestural impulses and task execution to be retained. Preserving the link between instinctive action and task has enabled this study to produce syntax that is intuitive and versatile.

## 6.4 Ideal interface configuration

To identify the ideal mode of machine interaction a range of interfaces with differing configurations have been explored. These include the fully unencumbered interface, which allows people to interact with computers using free hand gestures; the semi unencumbered interface, such as the multi-touch interface; and conventional interfaces such as the keyboard and mouse. Differences in the user to interface configuration between each of these interface models have been noted. These differences demonstrate that two styles of user interaction - face-to-interface and outward-facing - exist within the broader UGI paradigm.

#### 6.4.0.4 Unencumbered gesture interface

The unencumbered gesture interface model depicted in table 6.1 (see page 122) represents the model of interaction developed during this research investigation. The Gesture-Face Layer (Chapter 3, pages 71 - 56), Gestix interface [Stern et al., 2006] and VIDEOPLACE [Krueger et al., 1985] are all fully unencumbered models of interaction that are mediated by optical gesture recognition. These models represent face-to-interface interaction between people and digital interfaces.

#### 6.4.0.5 Semi-unencumbered

The semi-unencumbered interface represents a mode of interaction that moves away from the current model of HCI, where interaction is mediated through peripheral devices that are most often mechanical in nature, such as the mouse and keyboard. Semi-unencumbered interaction is unencumbered in the sense that it does not require peripheral input from mechanical devices, however this model still requires that the user remains proximal to the physical interface. Through this research two semi-unencumbered models of gesture interaction have been identified. Though these devices are both semi-unencumbered there are notable differences in the syntax and user-to-interface configuration. For example the semi-unencumbered model depicted in table 6.2 (page 123) is a tactile-interface where the user faces the interface in a traditional face-to-interface workspace model. The model depicted in table 6.3 (page 123) is a wearable interface, which

is worn by the user. The workspace composition of this interface does not follow the usual model of face-to-interface interaction. Instead, this interface has an outward-facing composition, where both the output is projected and input is captured from in front of the user.

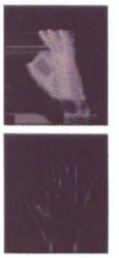
#### 6.4.0.6 Tactile-Interface

The multi-touch interface developed by Han [2005*b*] represents a semi-unencumbered mode of interaction. Though like the fully unencumbered employing gestures in face-to-interface style interaction, this mode still requires a user to physically handle the interface. As with encumbered keyboard interaction the users still need to be proximal to physical interfaces. Interfaces like the Apple iPhone and Microsoft Surface facilitate a similar model of interaction. These models of interaction are emerging as potential replacements to the current mode of keyboards and mice interaction. The success of these interfaces is due to the greater flexibility they can offer users and interface designers. As a result of not being constrained by similar physical limitations of hardware keyboards multi-touch screens enable greater customisation of the user interface, through the use of executable software keyboard applications. For example, applications can be developed to allow users the ability to alter a keyboards size, position and keypad layout. Such flexibility limits the potential for inefficient layouts to become embedded into popular usage. Furthermore, keyboard applications can be designed to aid the visually impaired, with the use of larger font sizes. The most significant aspect of this interface model is that it enables all of these parameters to be defined by the user. Not only does this have the potential to massively increase a users ability to customise their interfaces it can also reduce the economic impact of manufacturing physical keyboards. As a consequence of these innovations, multi-touch screens will likely alter the design of future machine interfaces, particularly those used on small handheld mobile devices.

#### 6.4.0.7 Wearable devices

In table 6.3 an outward-facing models of gesture interaction is depicted. Wearable interfaces such as the gesture pendant developed by Gandy et al. [2000] and the Sixthsense device developed by Mistry and Maes [2009] represent a third model of gesture-interaction. These devices represent outward-facing models of gesture interaction, as the device faces outward acquiring a similar visual perspective as the user. In the case of Mistry and Maes [2009] Sixthsense interface the device is capable of augmenting virtual objects on to real space through the use of an optical projector. Though they may facilitate free hand interaction, wearable devices still represent forms of encumbered interaction, due to the interface being worn by the user. Consequently the space between the user and the interface is fixed and inflexible reflecting a degree of rigidity within the syntax of interaction.

Table 6.1: UG interface

| 6.1a | 6.1b | 6.1c |
|---|---|---|
|  |  |  |
| Unencumbered computer facing gesture interaction | Enabling iconic deictic and ergotic interaction | Tracking using motion, depth and templates |

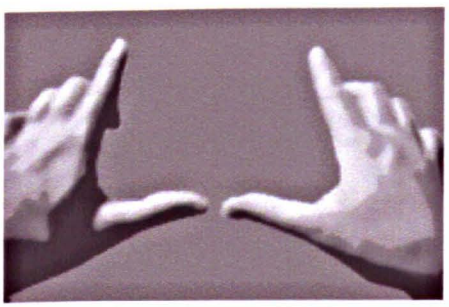
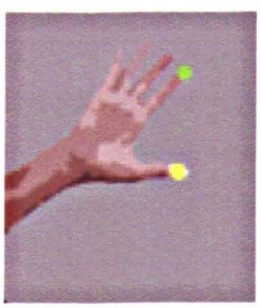
#### 6.4.0.8 Future interfaces

The three models of gesture interaction discussed though similar represent distinct paths for future interface development. It is likely that each model has an important role to play in how people interact with the machines and environments of the future. Both multi-touch and wearable outward-facing interfaces are likely to figure prominently in the evolution of personal mobile devices. Minor changes to current handheld multi-touch devices would allow each interface to exist side

Table 6.2: Tactile interface

| 6.2a | 6.2b | 6.2c |
|---|---|---|
|  |  |  |
| Semi unencumbered interface | Deictic and semiotic | Tracking using motion and blob detection |

Table 6.3: Wearable device

| 6.3a | 6.3b | 6.3c |
|---|---|---|
|  |  |  |
| Outward facing gesture interface | Enabling iconic deictic and ergotic interaction | Tracking using motion or colour detection |

by side in a complementary manner. However, unlike multi-touch interaction the outward-facing devices have limited application outside the context of small mobile devices. This limitation results from the fact that input is captured from the perspective of the user. The only position where such an interface would be able to recognise the gestures illustrated in table 6.3 are from around the users neck. The multi-touch interface shows greater promise of versatility than the wearable interface when it comes to the range of potential applications it could be utilised with. For example, in addition to being used on mobile devices it can be used on desktop interfaces and any machine requiring tactile-user interaction. Due to the fact that it offers greater flexibility than the mechanical keyboard and mouse it

is likely to supersede these devices and usher in a new paradigm of gesture based interaction. The fully unencumbered interface advocated in this thesis also has a significant role to play in the evolution of computer interaction. As computing becomes ever more pervasive and entwined into daily human activities, interfaces will be needed to support a diverse set of activities. New interfaces capable of providing comfortable and efficient interaction with robots, virtual environments and machines will be needed as alternatives to cumbersome and unsuitable interfaces such as the mechanical keyboard and mouse. To enhance our interaction with robots it would be useful if they could respond to visual cues such as gestures. To interact with and manipulate virtual objects with no physical surface capable of responding to touch, a method for recognising ergotic actions will be needed. Though currently the defacto interfaces used in HCI are the keyboard and computer mouse, these two devices are only able to facilitate single user interaction limiting the potential for people to collaborate in GUi environments. Currently the potential of the multi user interface is being hindered by the nature of the physical interface, as it is difficult for users to access keyboards, mice and joysticks simultaneously. Despite current limitations users demonstrate a willingness to engage in collaborative activity, through using the Internet, email and computer gaming. However, the entire architecture of modern computing is based around single user interaction. The unencumbered models of interaction discussed offer the potential to facilitate multiple user interaction, as users will no longer be restricted by the composition of the physical interface.

## 6.5 Conclusion

As computing becomes ever more entwined into daily human activities, new interfaces will be needed in order to support interaction with robots, virtual environments and real objects. The increasing viability of UGI is demonstrated in the emergence of commercial interfaces such as those developed by GestureTek, Xbox 360 and Sony eyetoy. These interfaces demonstrate that UGI has the potential to be both commercially successful and widely adopted. However, the success of QWERTY and WIMP interfaces should demonstrate that once an interface is

widely adopted by users it has the potential to become embedded into subsequent developments. Though evidence showed that QWERTY might not have been the most efficient interface available, persuading established users to utilise alternatives, such as DVORAK, proved problematic [David, 1985; Norman and Fisher, 1982]. Being the first typing interface to gain widespread distribution resulted in the QWERTY layout gaining dominance in the HCI paradigm, allowing word-processing to become synonymous with the QWERTY interface in the minds of most computer users. As a consequence, this keyboard layout is widely manufactured reflecting what David [1985] describes as path-dependences. Therefore, it is important that interface developers understand the potential consequences of introducing particular interface syntax. The evaluation framework and method developed through this research goes some way to help future syntax developers recognise how gesture syntax can impact users. Prior to the recent expansion of the domestic and personal computer market, investment in computer technology has predominantly come from the advancement of telecommunication in business and military sectors. As a consequence, developments in software and hardware have likely been driven by concerns such as price and productivity. These factors would have had the potential to initiate the cycle of path-dependences. Despite the strong evidence that demonstrated the DVORAK interface offered a 10 percent efficiency gain [Norman and Fisher, 1982] , the costs of reequipping businesses might have acted as a prohibitive factor in decisions to adopt alternative technologies [Liebowitz and Margolis, 1995]. Currently, the expansion in domestic and recreational computer usage is beginning to become a significant factor driving innovation. The influence of the end-user is increasing in parallel with the development of novel technologies. The newly emerging relationship between technology and end-user is likely to change the way that path dependences occur in future development cycles. The transition and emergence of new HCI paradigms are also likely to be more fluid as investment in innovation becomes distributed among a wider user base. One need only look at the popularity of Apple app-store to see how the relationship between user and technology is beginning to change. Not only are users playing a greater role in determining the success of an interface, the pool of developers contributing to the design of current interface applications is also growing. The business model adopted by Apple app-

store seems to reflect the suggestion that end-users represent much more flexible decision makers than businesses or industry. This is probably due to there being greater risks associated with bulk purchases and the higher levels of investment a business would be exposed to when making acquisitions. Aided by this increased flexibility of technological development and user adoption cycles, developers will be much more able to put the concerns of the end-user at the heart of interface developments. However, in order for the field of UGI to develop ergonomically and avoid the paths taken by previous interfaces clear principles and guidelines need to be established. Throughout this research a clear methodology has been demonstrated and revisions to methods and practices have been advocated. In considering whether current methods and practices used by developers and evaluators of encumbered HCI are relevant to UGI, it was found that though the underlying principles are relevant a range of revisions are needed. Firstly, the development of a robust set of general guidelines for inspecting the efficiency of gestures would greatly benefit UGI developers. The creation of an open iterative guideline framework that allows multiple evaluators to assess each others findings may produce a universal set of standards that allow UGI developers to be confident about how efficiently their syntax performs. Significantly, understanding the biomechanical constraints of people is even more critical to UGI syntax development than it is to encumbered forms. As the gestures utilised during interaction with encumbered interfaces are constrained by the physical mechanics of the interface, developers can be confident about how the syntax of an interface will impact users. Whereas the unencumbered gesture interface has no mechanism to constrain a users actions. Therefore, it is important that developers of syntax completely understand the strains each gesture will place on interface users. Finally, the unencumbered interface requires a more complex evaluation model as the mechanisms for controlling the interface are hidden. Consequently, there are additional variables to consider when examining UGI syntax, such as a users interpretation of the syntax and the potential for variation when specific gestures are replicated. Alternative approaches to gesture syntax development have been discussed in this investigation (see chapter 5, page 109). However, these approaches rely on syntax derived from 2D image templates. The inherent limitation of this method means only partial recognition of human gesture is

possible. Such methods make little distinction between the use of semiotic and ergotic gestures, and generally reduce all actions, irrespective of type, into a single image of a static posture. As semiotic gestures can convey meaning through hand shape alone, they can effectively be recognised using the 2D image maps. However, ergotic gestures cannot consistently be recognised with this method, as the trajectory and context behind an action is important to identifying a users intention. As a consequence, semiotic actions are more widely use in gesture syntax development. This investigation recognises that there is a clear distinction between the use of semiotic actions to convey ergotic impulses and the use of ergotic action to complete ergotic tasks. By utilising four-dimensional (4D) image templates derived from depth and motion disparities (Chapter 3, pages 58 - 66), this study has been able to develop a reliable method for recognising a entire phase of a gesture. As a consequence a broader range of gestures can be successfully recognised and a more robust and versatile interface syntax can be created. In the course of this research different gesture syntax have been identified. These include natural gesture, signed language and syntax created specifically for computer interaction. The syntax of natural gesture has evolved together with speech to utilise similar sensorimotor apparatus. Formal signed languages, such as British Sign Language (BSL) and American Signed Language (ASL), have also evolved in parallel to speech. However, these syntax models have a clearly defined syntax that is illustrated in HA/TAB/SIG/DEZ stokoe notation model. This research has found that the physical exertion required to communicate using sign language would offer no extra physiological benefits to the user than current models of interaction, such as keyboard and mouse interaction. Therefore it is important that a distinction be made between using established gesture lexicons - like BSL and ASL - and developing syntax that facilitates ergonomic computer interaction. To advance the ergonomic application of gesture in computer interaction a specialised syntax that reduces user exertion is required.

### 6.5.1 Further questions and future work

Though all of the aims and objectives of this research have successfully been addressed, there are still some significant questions that would benefit from further exploration. While this research produces a dataset that enables both user preference and computer recognition accuracy to be collated through a single archive these calculations are based upon the use of static representations of gestures and do not include motion history volumes (MHV), which depict the full phase of a gesture as a 4D template in a single image map. Future research will work to incorporate 4D image templates in the compilation of the GEF dataset. The use of static representations of gesture does not however diminish the conclusions of this research particularly in relation to the physical preferences of users. Furthermore despite the successes this research has not implemented a method for evaluating the memorability or the intuitiveness of a gesture lexicon. These are issues which will require further investigation at a later stage. This should not however diminish the fact that through this research a set of hand postures can confidently be defined as adhering to generalised comfort thresholds. There are three other areas identified through this research that would benefit from further examination. First, how will augmenting computer graphics to users physical actions affect the quality of users responses. Second, can the use of graphic processing units (GPU) improve a computers ability to learn optical information. Third, can monocular lenses that are widely distributed throughout the consumer market be adapted for stereovision. This research suggests that these issues will be important to the future success of unencumbered interaction. As the realism of augmented graphic responses to a users action is likely to be a significant factor in attracting users to future interfaces the speed of graphic processing may be considered crucial to success. A delay in rendering of virtual objects has the potential to frustrate and disorientate users. For instance, when using a computer mouse it is useful if the onscreen cursor responds in as close to real-time as possible. Limiting the potential for a time lag between the users actions and the graphic response will enable users to better coordinate their actions with the graphic interface. At present, the increased computer-processing load required for a computer to detect a gesture can produce a sufficient enough delay as to confuse users, subsequently impact-

ing user experience. Recent developments make it possible for computer graphics cards to be programmed to carry a significant portion of the data processing load [Farrugia et al., 2006; Viney and Green, 2007]. Such advancements are likely to increase the speed with which computers are able to render graphical data. Such developments could potentially lead to the augmented real and virtual becoming increasingly indistinguishable in the minds of users and architecture of computers. For example, the parallel processing of graphics and data will likely enable computers to produce increasingly responsive graphics that may allow a user to suspend disbelief in the artificial nature of augmented and virtual reality. The specialised graphics-processing unit (GPU) will enhance a computers ability to simulate the outcome of events in a manner akin to how the mirror neurons operate within the sensorimotor processing apparatus of humans (Chapter 2, page 40). Such apparatus are not only critical to how we perceive our environment, but are also critical to how we learn and interpret both manual and aural languages. For instance, a computer equipped with a dedicated graphics-processing unit will be able to treat the input of a camera in the same manner as a graphics simulation. A computer programmer would be able to write object-orientated code that could treat both simulated and optical input in the same way, increasing the abilities of computers to learn cause and effect by running simulations based on real world optical input. Though there is nothing preventing programmers using this method the use of graphics-processing units would encourage this as a default position. These developments have the potential to bring the fields of computer-vision and computer graphics closer together to create a hybrid field that focuses on optical cognition. The development of such a field is likely to increase the viability of UGI systems and further advance the development of artificial intelligence, as a consequence focussing on this topic could prove a rich source for future research. In this research the optimal configuration of an optical gesture interface has been identified as requiring a combination of motion segmentation, depth disparity image maps and motion history volumes. The common approach to creating depth disparities is to implement stereovision and the use of two cameras that can be calibrated to facilitate stereopsis. The problem with this approach is that it relies upon the end users either possessing two identical cameras or a binocular camera. The obvious solution to this problem is for the users to purchase additional

equipment. Alternatively, a method could be developed to enable depth metric information to be obtained from a single CMOS or CCD image sensor chip. As it is relatively easy to program computers to interpret camera output, designing lenses that can produce varying degrees of parallax on a single image sensor could be a potential solution to equipping ordinary cameras with the ability to facilitate four-dimensional gesture recognition on generic consumer devices. Though UGI is an emerging field that relies on the development of cutting edge research and programming, there is little reason why this model of interaction should not be accessible to all computer users. Provided that a formal and universal ergonomic gesture syntax is established, UGI has genuine potential to reduce the repetitive strain injuries, increase productivity and improve the ease with which interface users can interact with electronic devices. Furthermore as this new interface can be implemented on existing consumer hardware, in addition to being economically viable and ergonomic, it should prove to be a more ecologically sustainable model of computer interaction, as the requirement of having to use peripheral devices may be completely eliminated.

# Appdx .1

## .1 Perceived physical exertion (ppe) of gestures

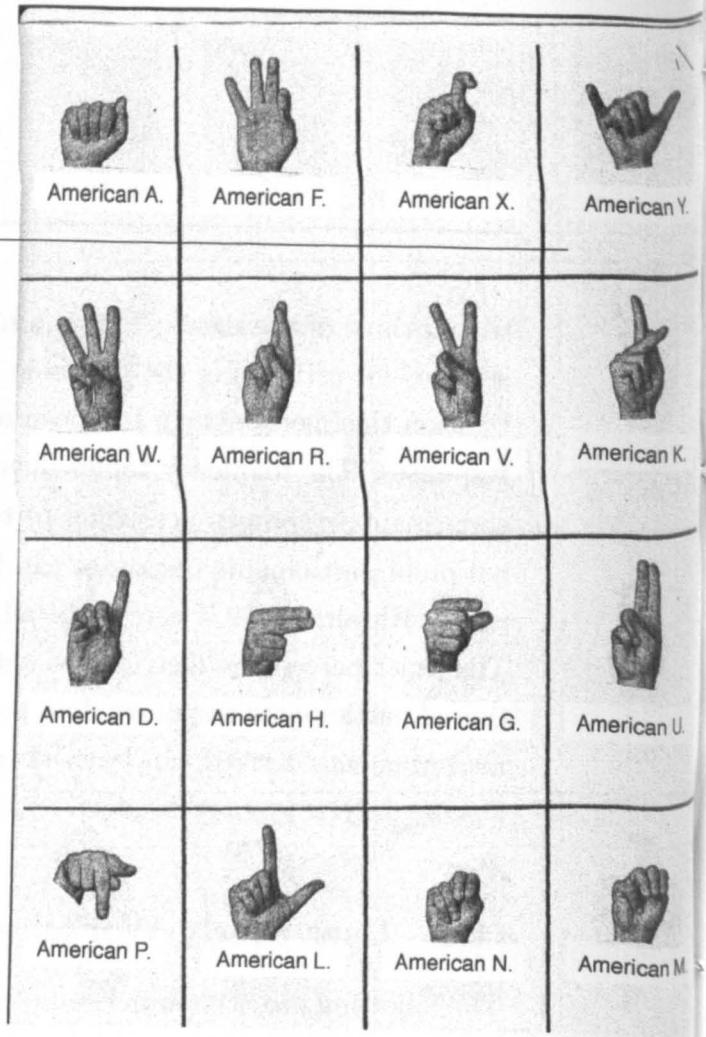
The measurements currently discussed represent the perceived levels of physical comfort and exertion participants experienced when replicating the postures included in the GEf dataset. To determine which postures can accurately be defined as comfortable a typical measure of each posture must be established. Dividing the overall sum of participants assessments by number of people in the sample produces a mean evaluation of the postures. However, establishing the mean assessment of each posture is not sufficient to accurately determine whether a posture can be defined as comfortable. The variation contained within participants responses must also be ascertained prior to establishing whether the mean accurately reflects the attitudes of the whole sample. Identifying the level of consensus that is present within the sample is significant to establishing an accurate measure of a postures comfort. The amount of consensus present within the sample has been calculated by finding the standard deviation within participants responses. By identifying the standard deviation within the sample it is possible to determine how opinions are distributed. Identifying the range of distribution enables this investigation to determine where there might be consensus and disparities in opinions. This enables the investigation to differentiate between conclusive results from those that are inconclusive. The results have also been checked to see if they adhere to the 68-95-99.7 rule, which is also known as the three-sigma rule. The principles governing this rule state that statistical consensus can generally be proven when all values contained in a set fall are

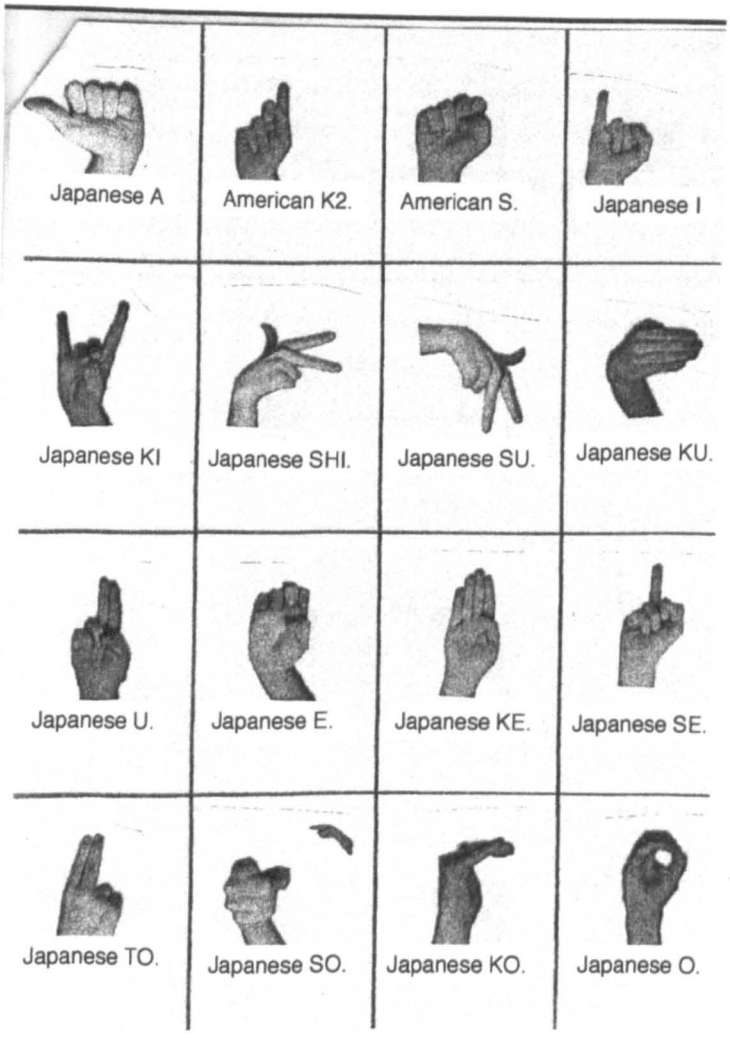
within three of the mean. This is a methodology that is utilised in the empirical sciences for estimating the probable accuracy of the mean result. In this investigation this methodology has been applied when evaluating participants overall responses. The frequency tables illustrated in figure 3.2 show the distribution of participants responses according to the most highly rated postures. The distribution of participants responses can be seen to roughly adhere to the 65-95-99.7 rules with almost 99.7 percent of all values falling three standard of the mean. The exact percentage distribution can be seen in table 3.1. Though the distribution of participants responses does not precisely adhere to the 65-95-99.7 rule the measurements derived can be used to identify postures that are skewed towards a general perception of comfort.

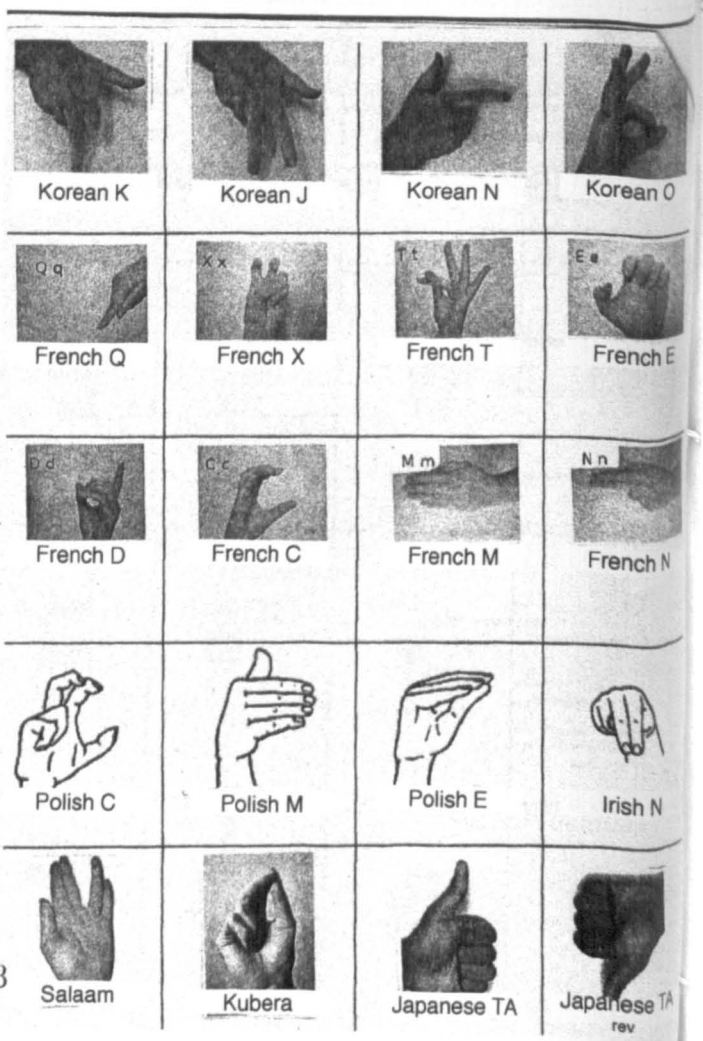
### .1.1 User Study Worksheet

The following pages contain examples of the questionnaire used in the User study documented in chapter 4 page 74

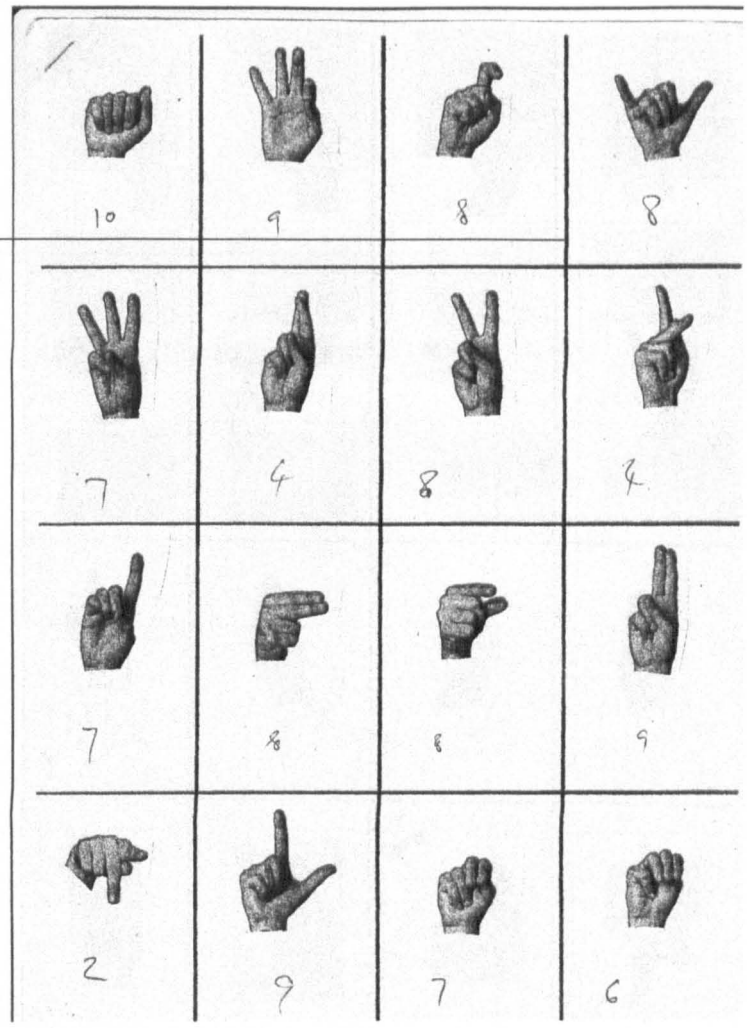
### .1.1.1 PPE Questionaire

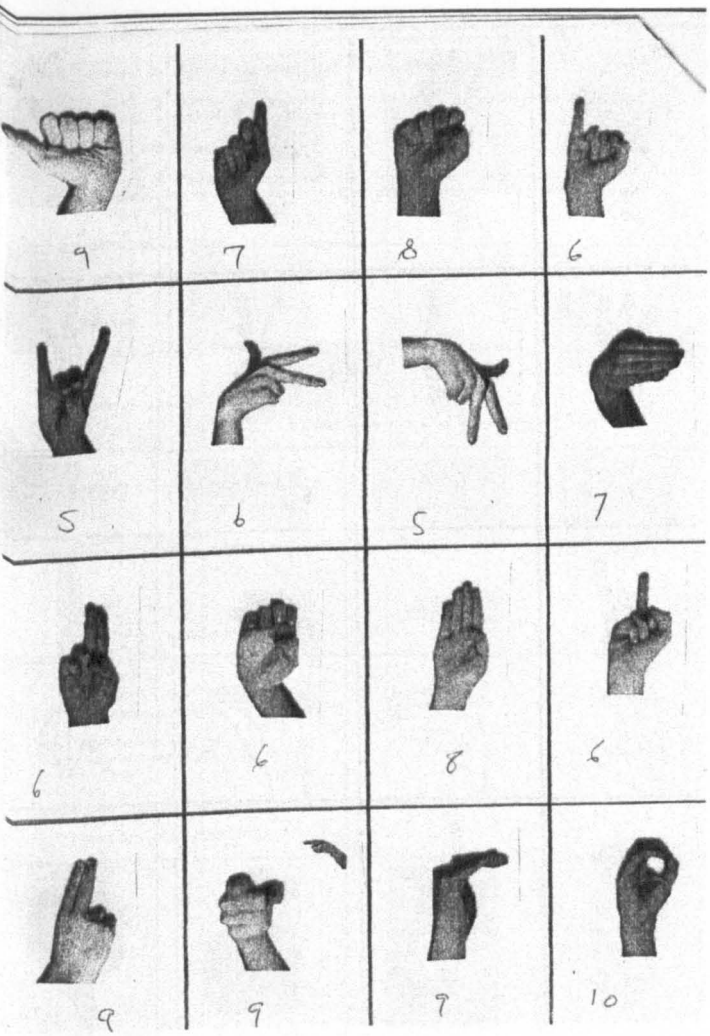


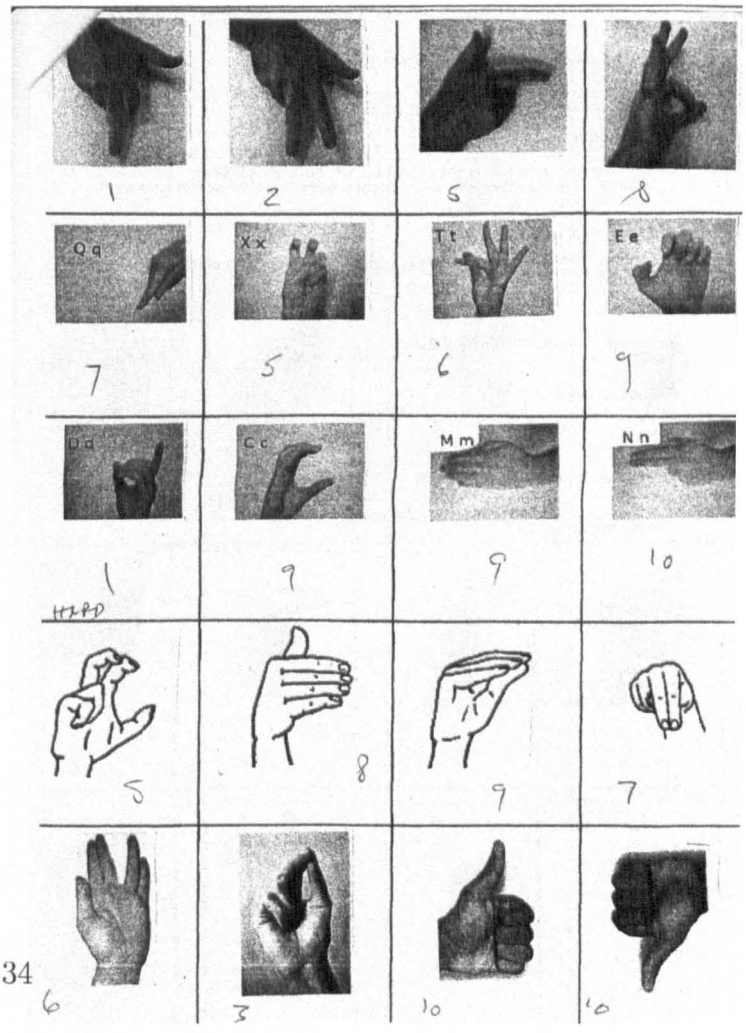



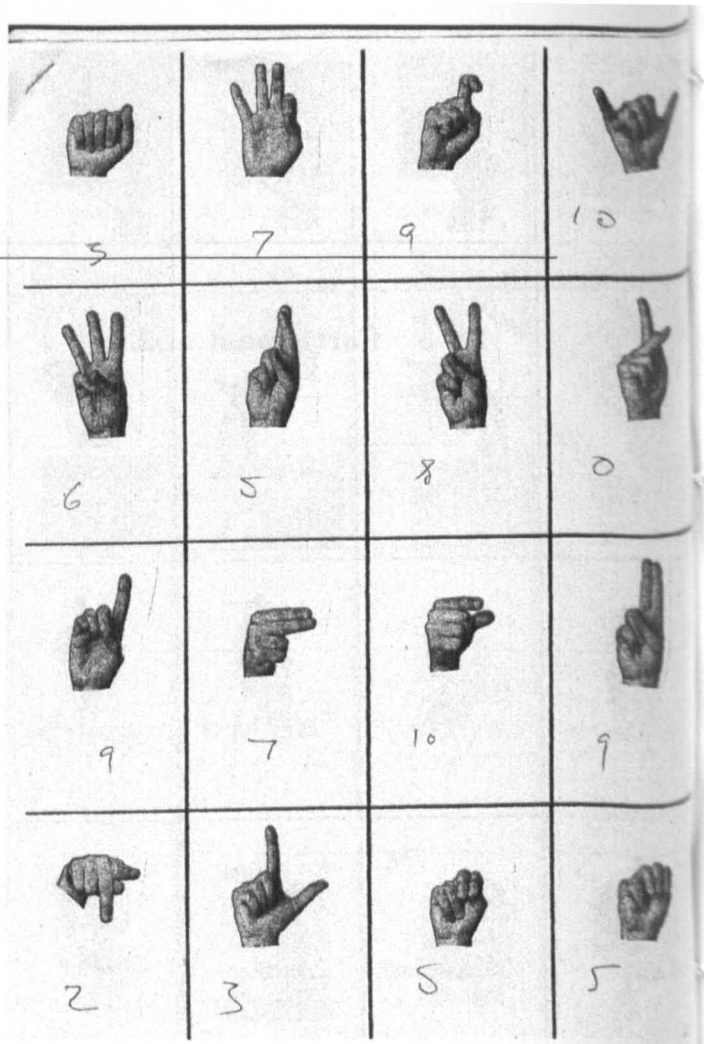
### .1.1.2 Participant ab2

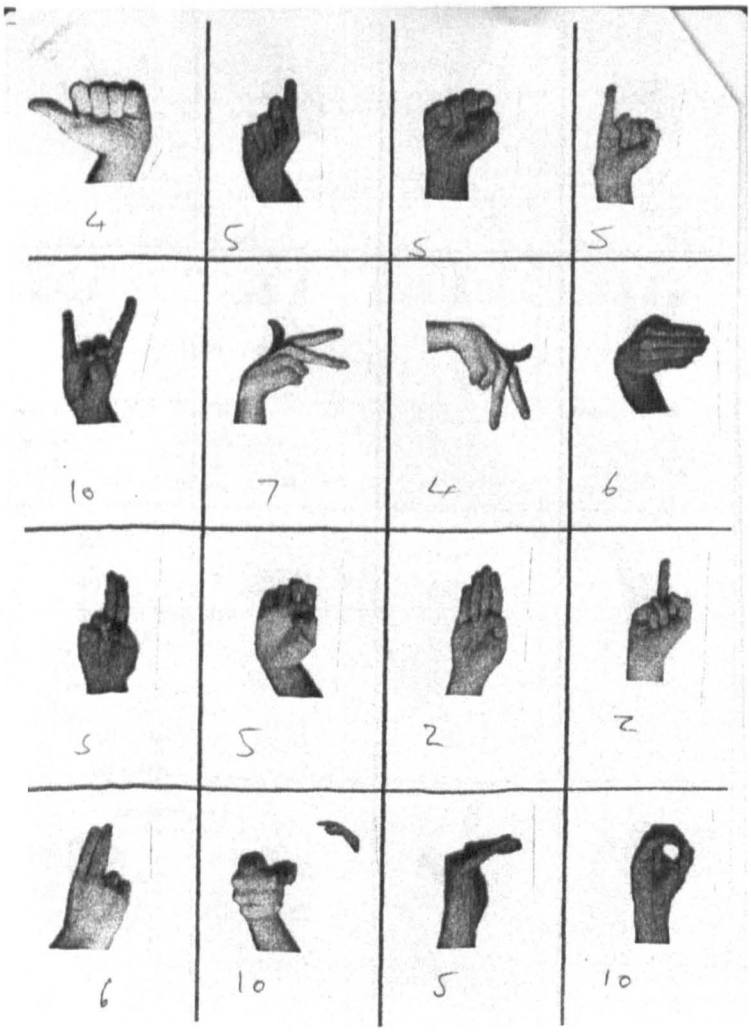


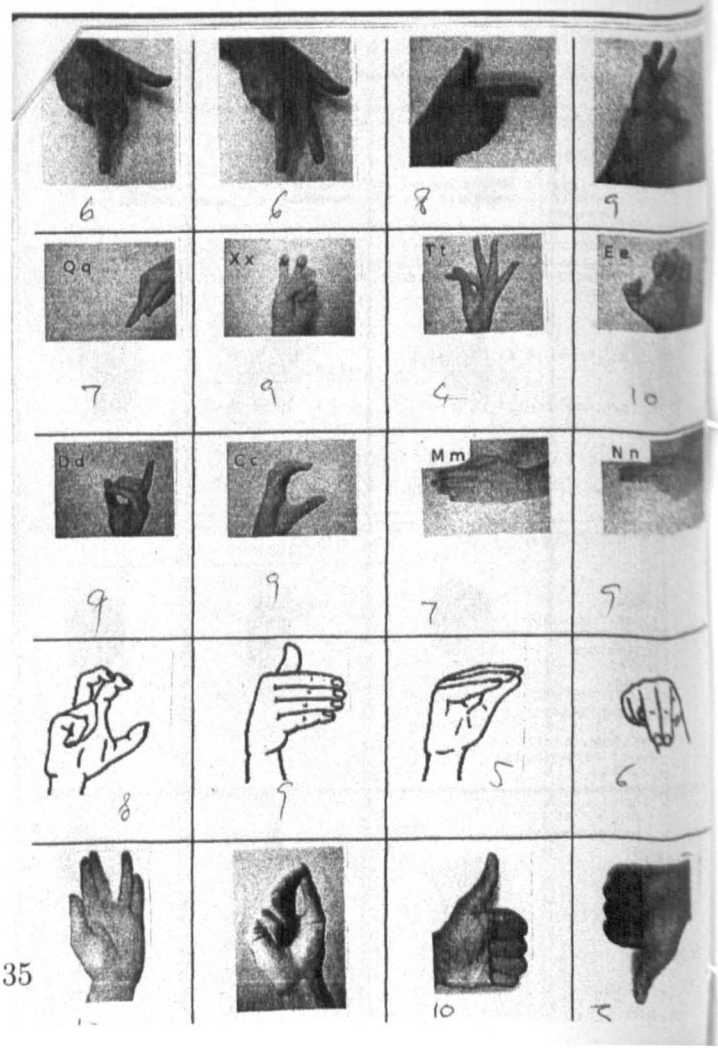



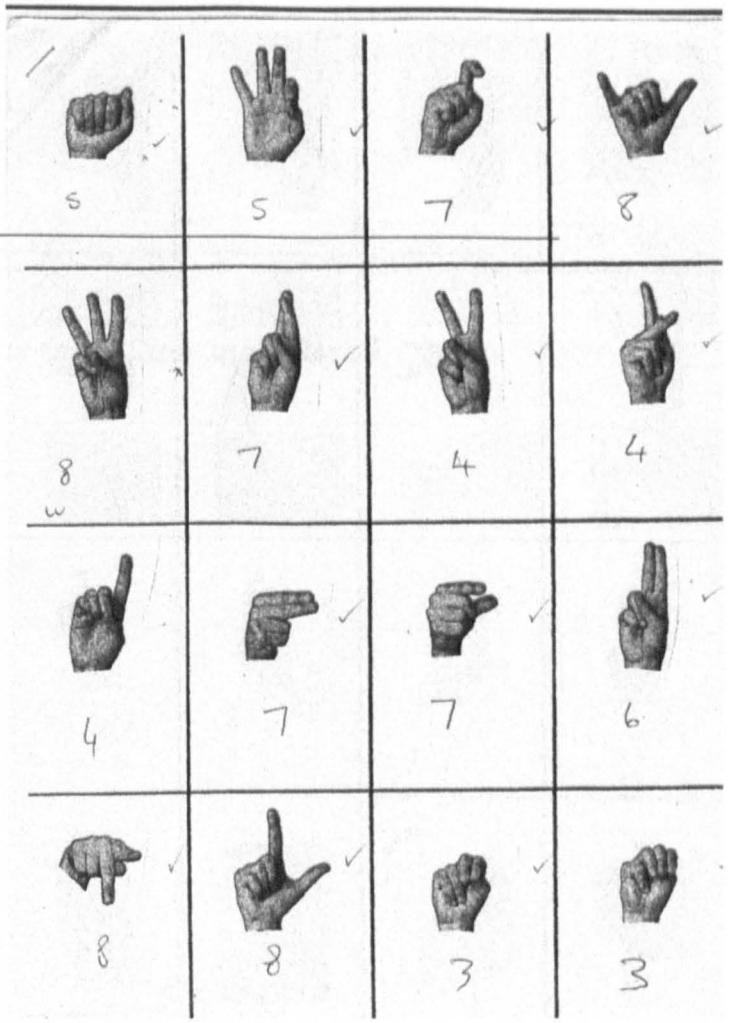
### .1.1.3 Participant ch2

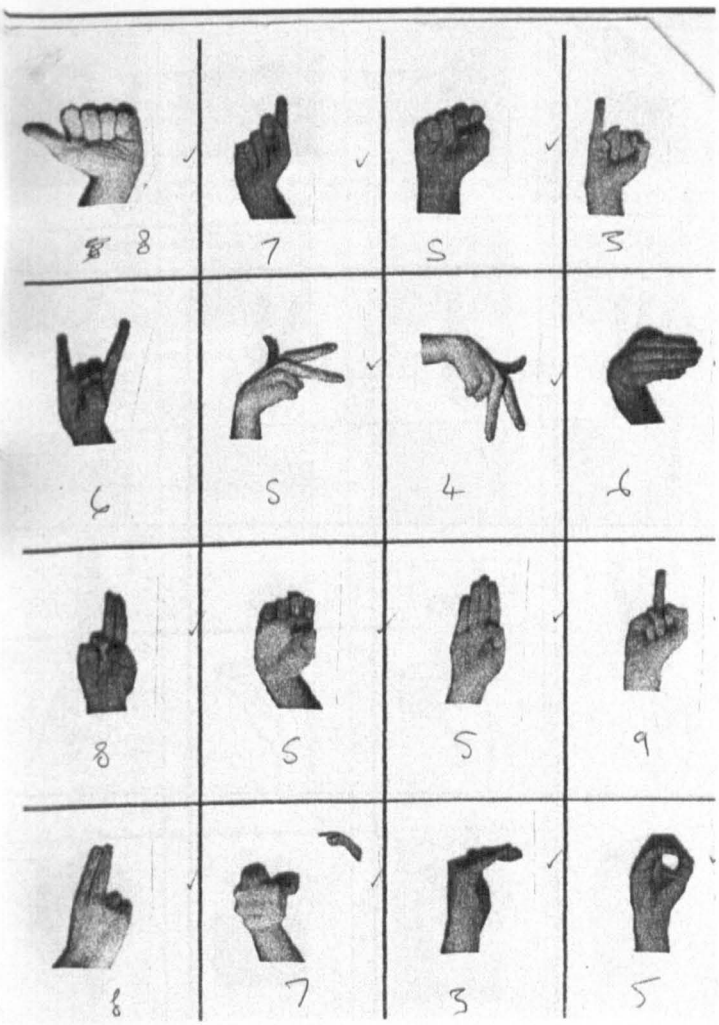


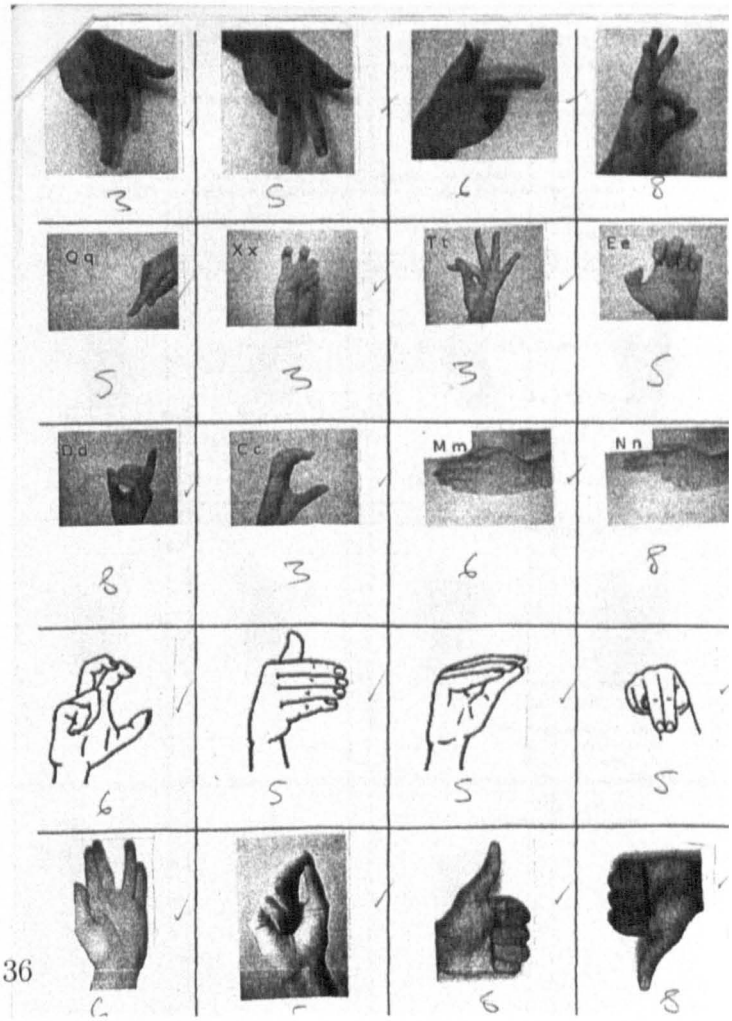



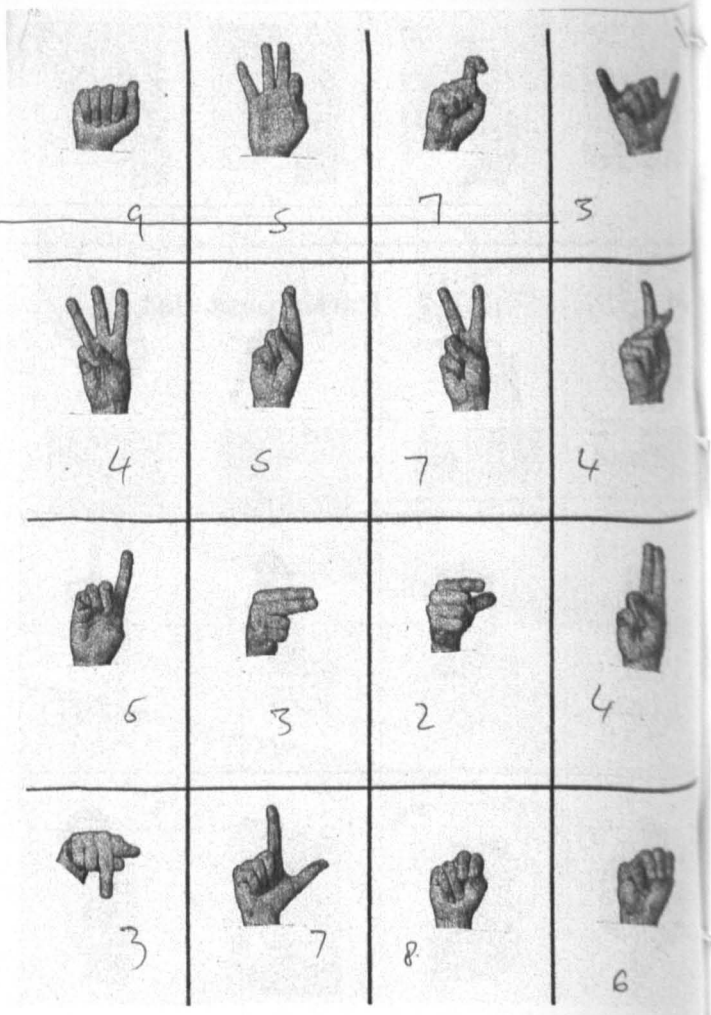
### .1.1.4 Participant da1

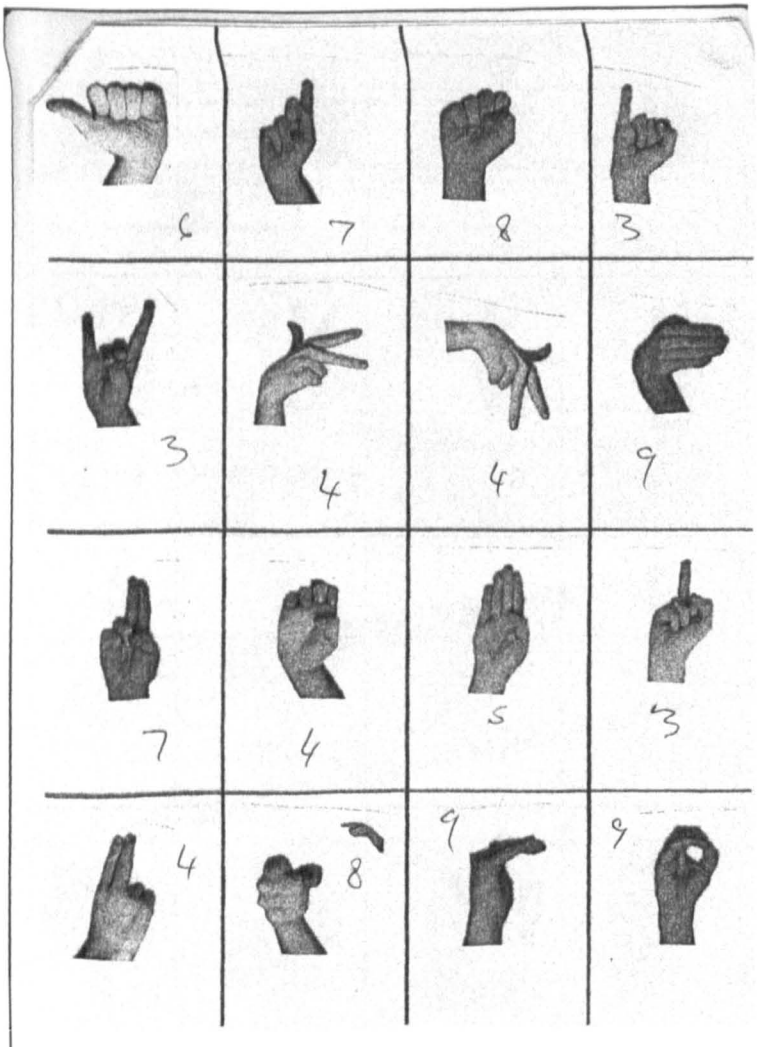


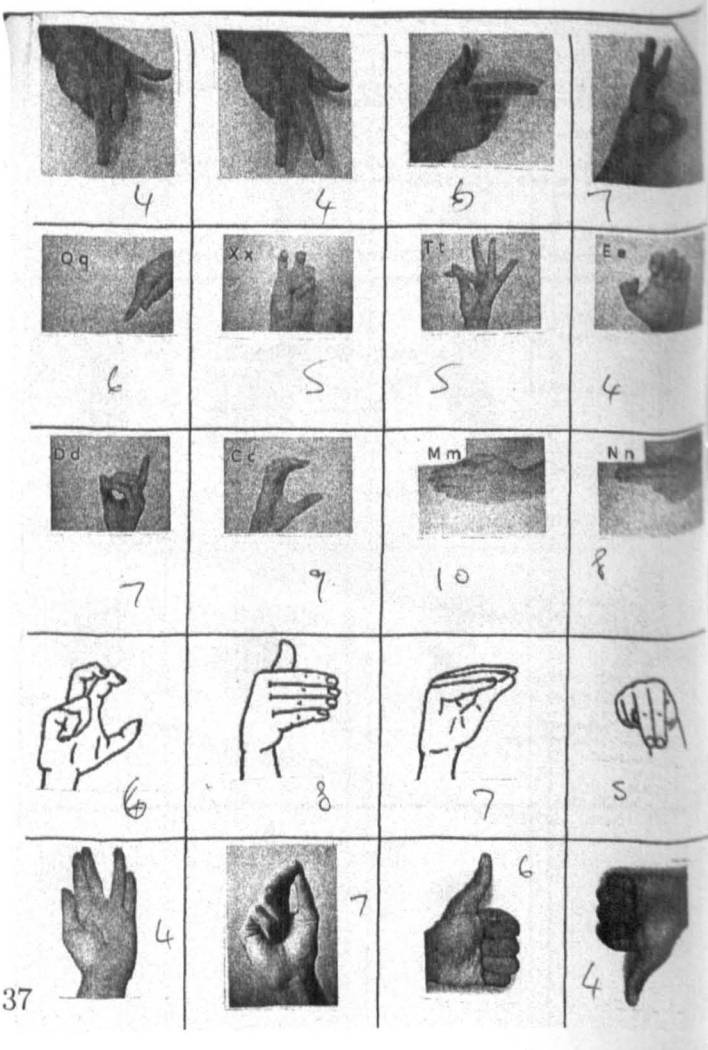




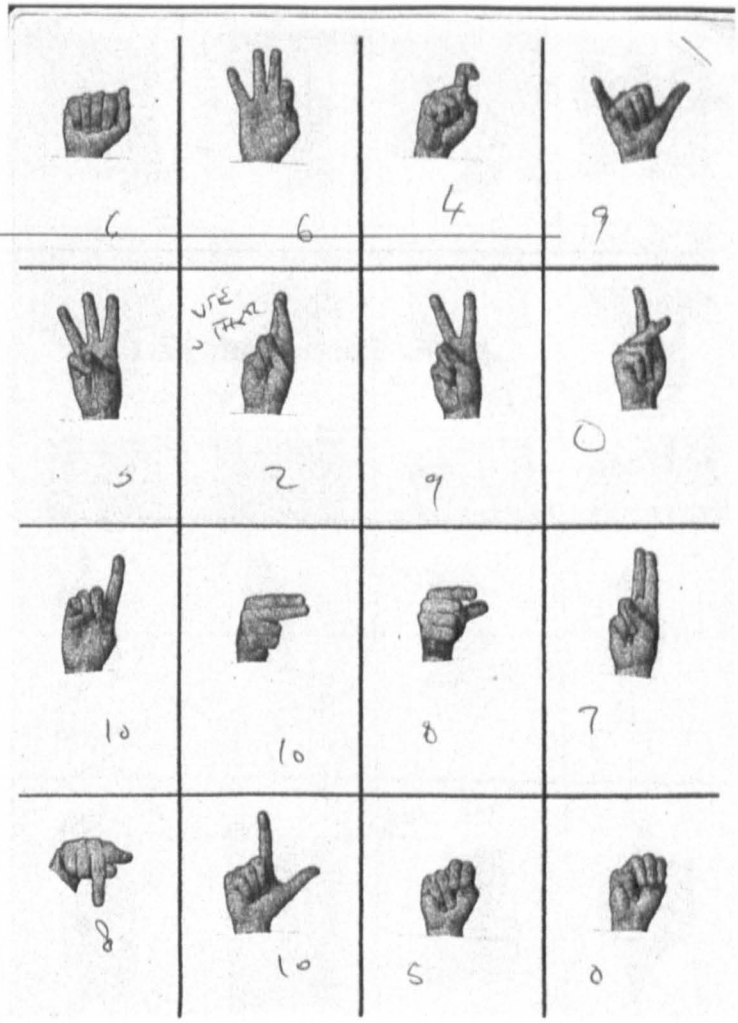
### .1.1.5 Participant em2

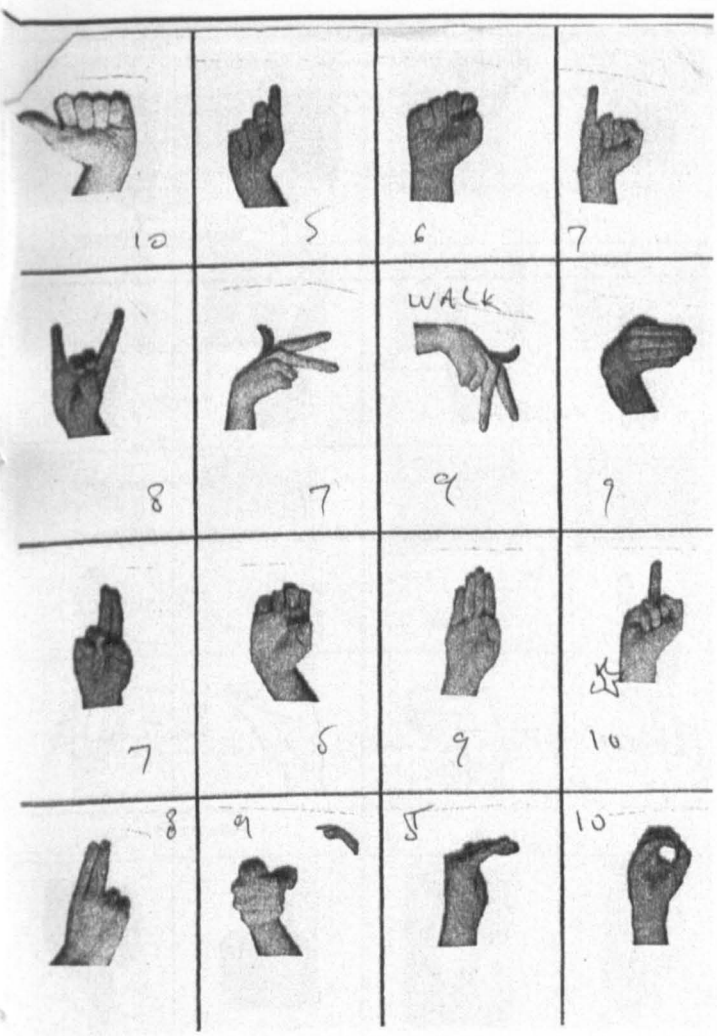


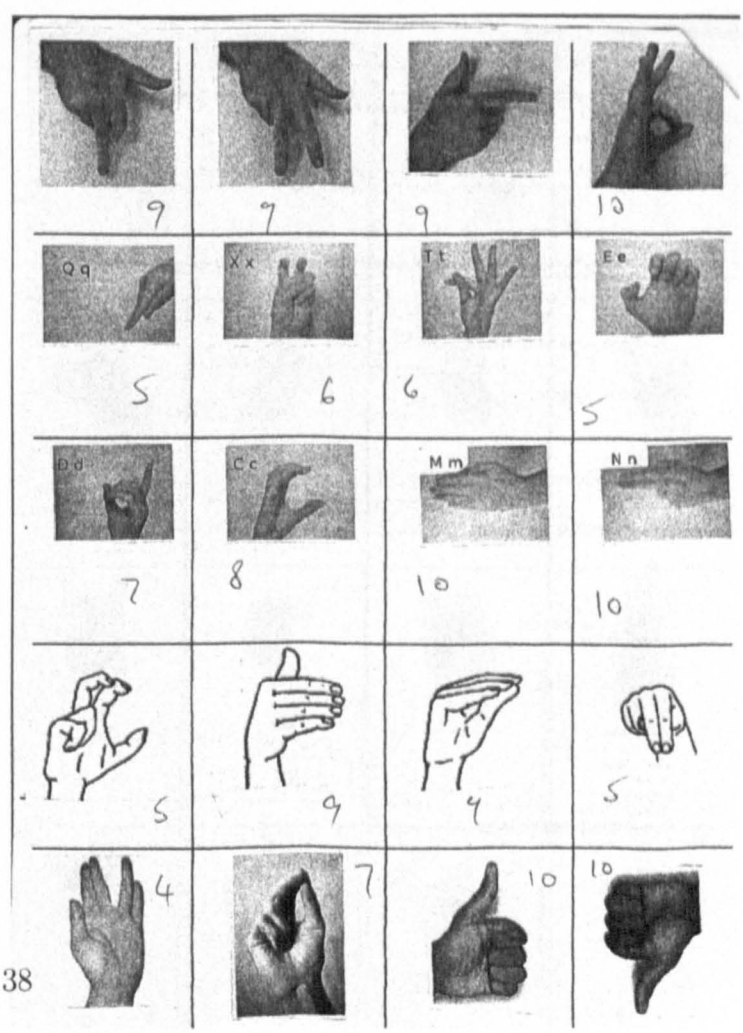



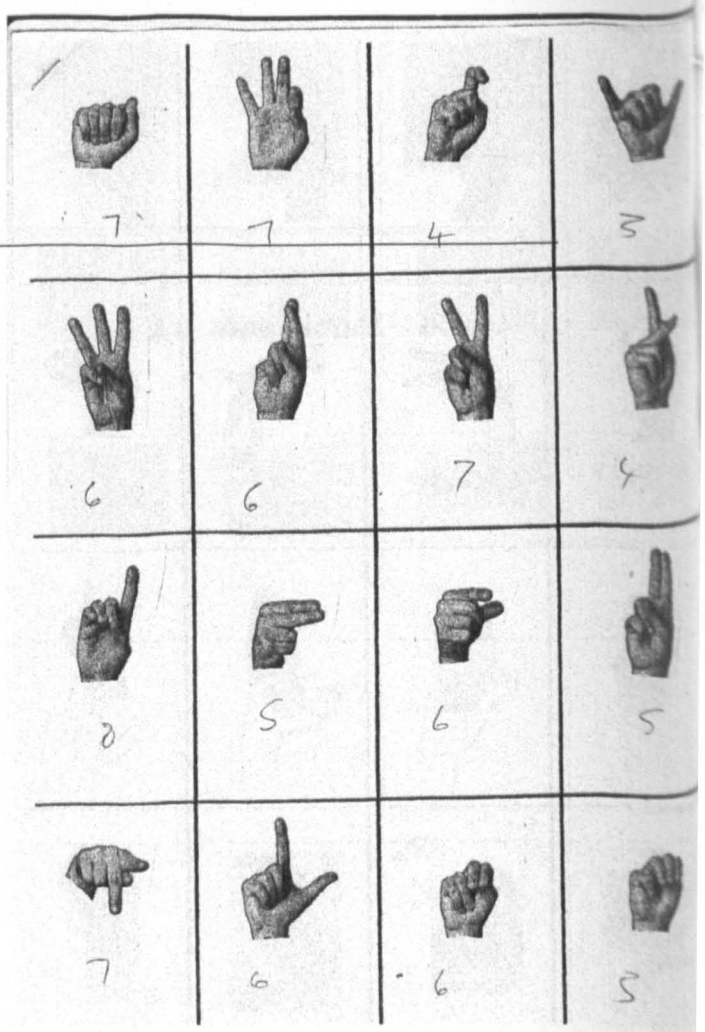
### .1.1.6 Participant fr1

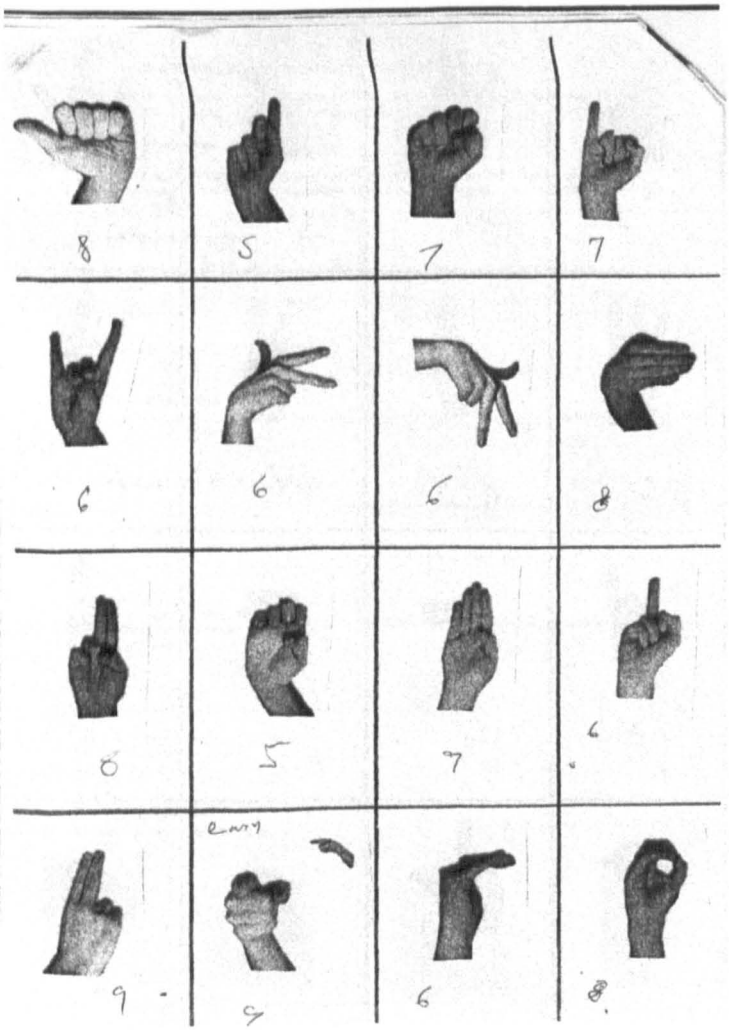


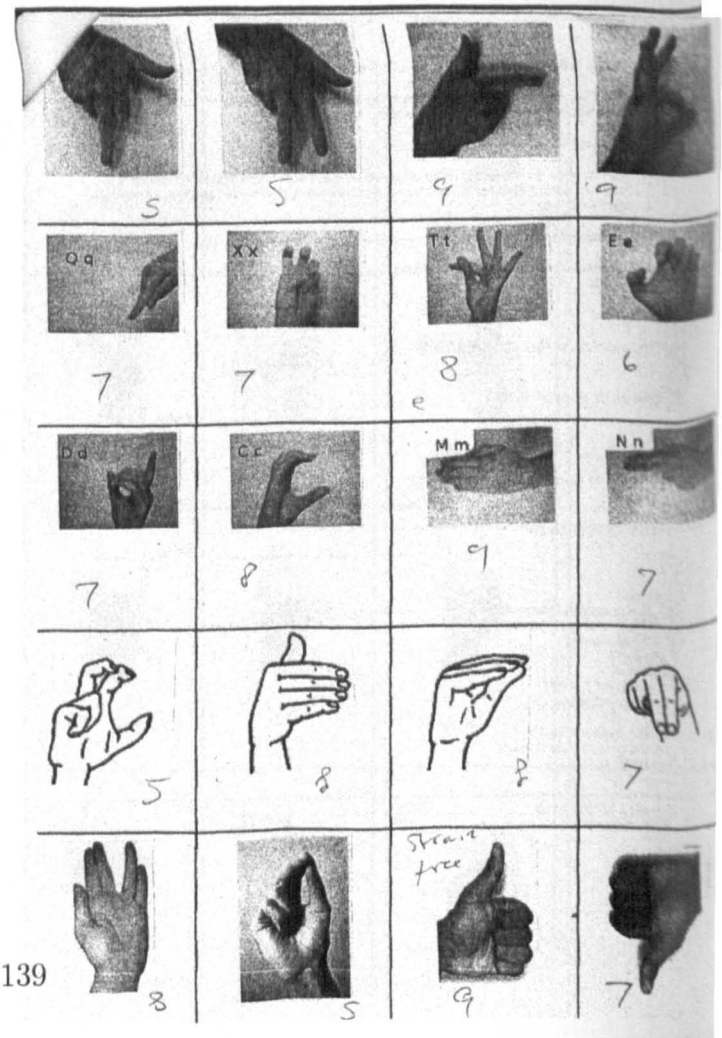



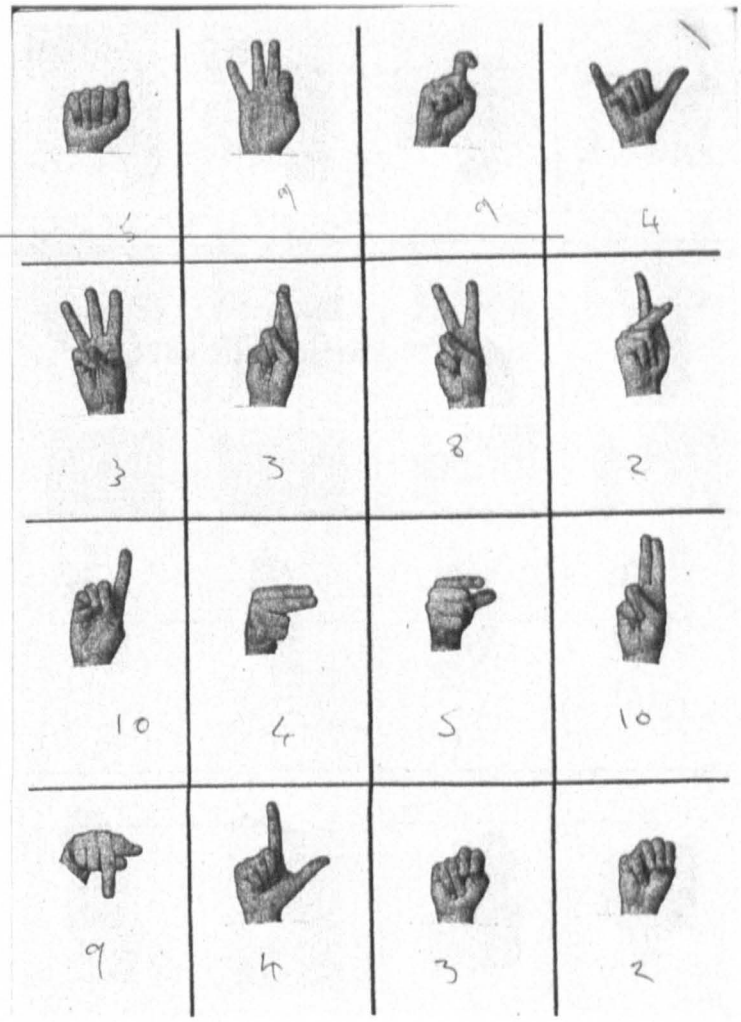
### .1.1.7 Participant jak1

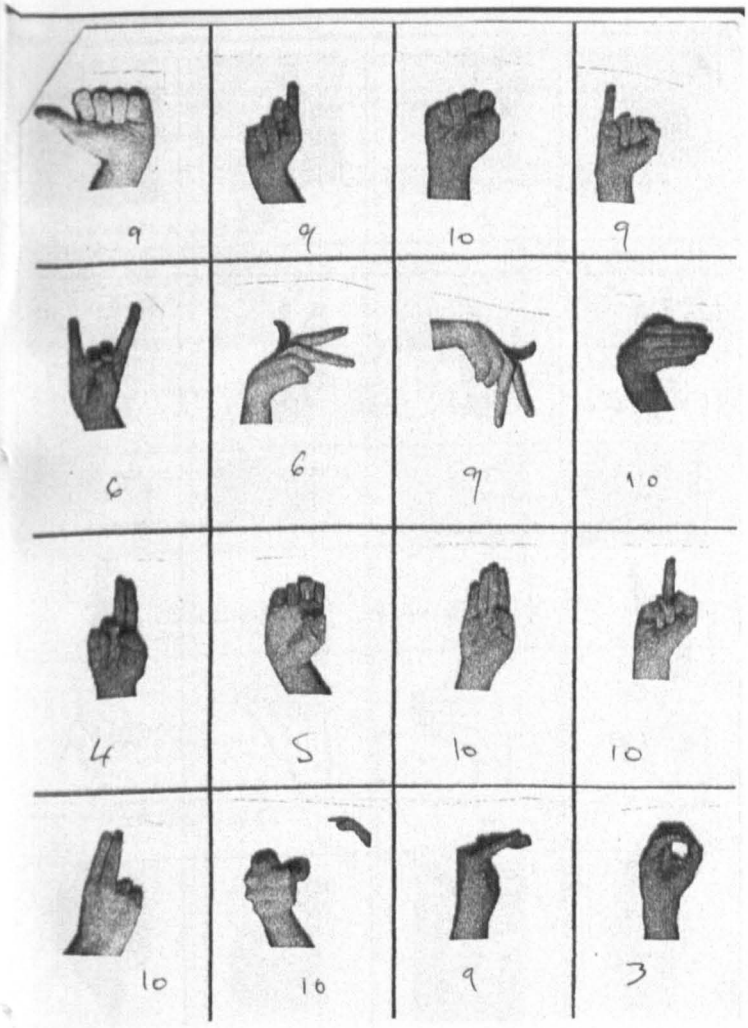


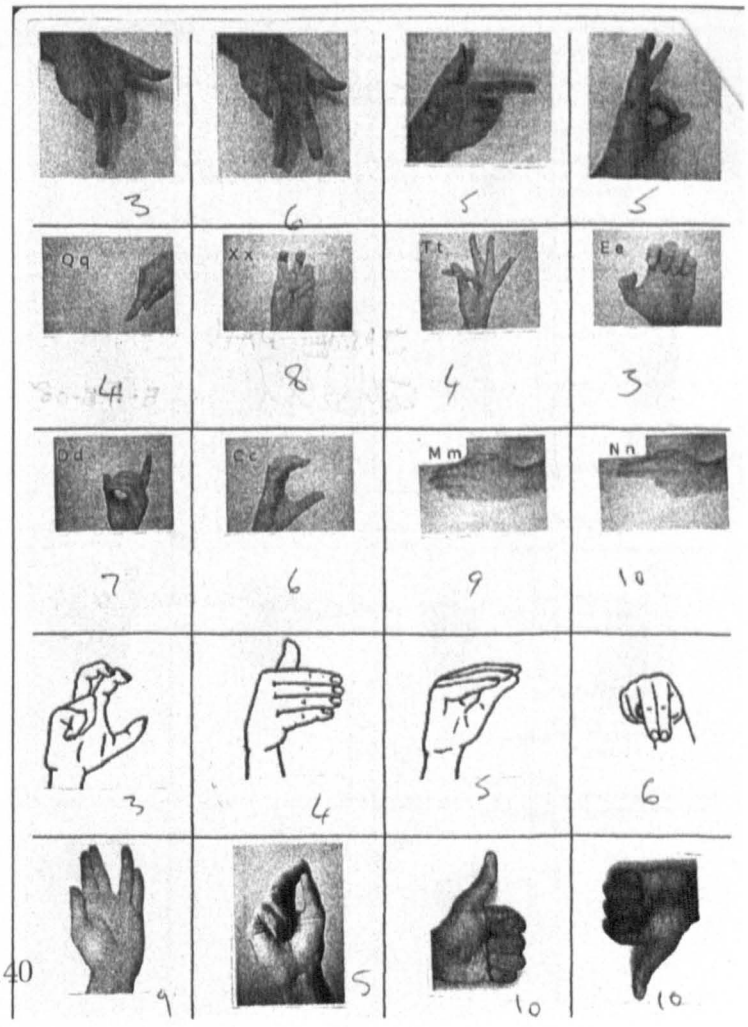




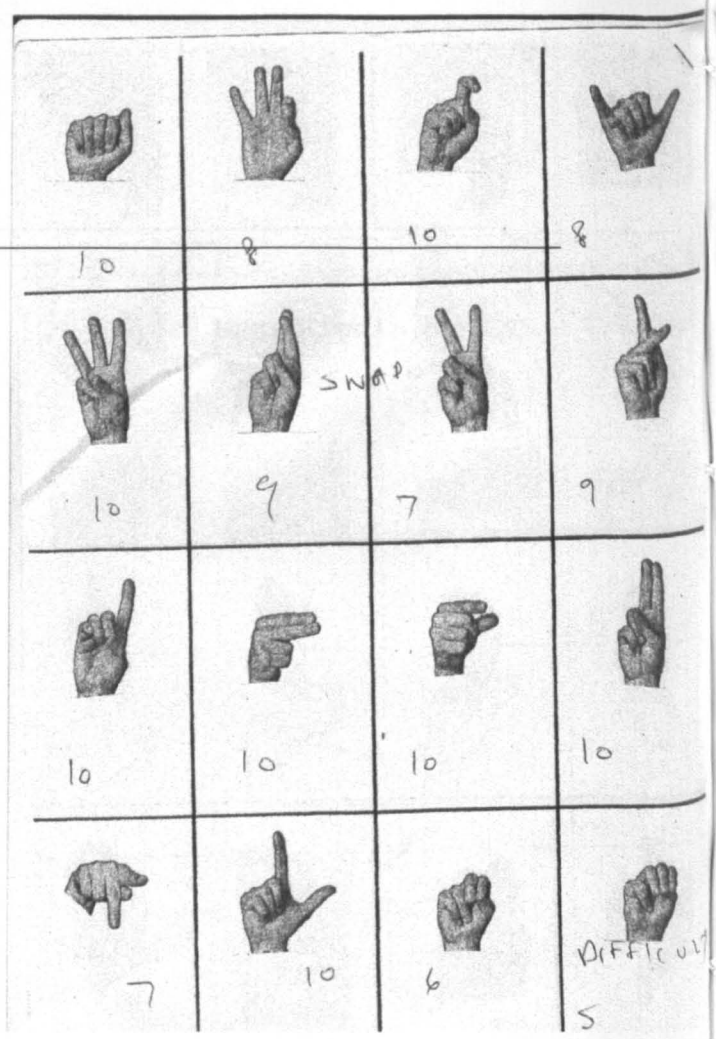
### .1.1.8 Participant ja1

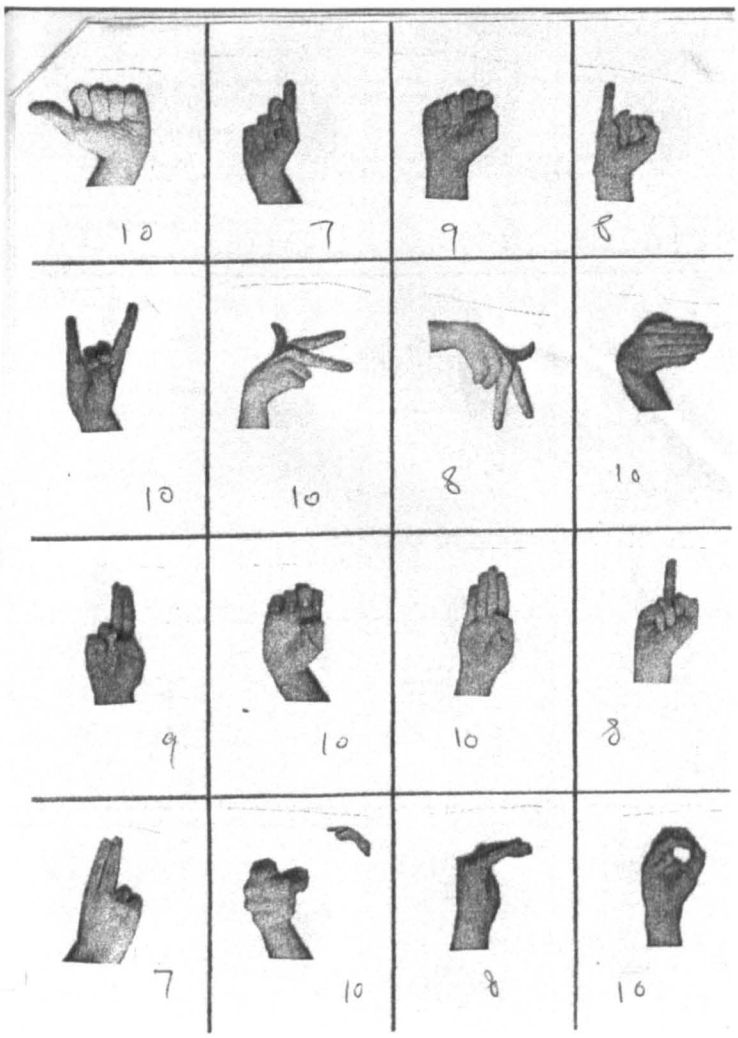


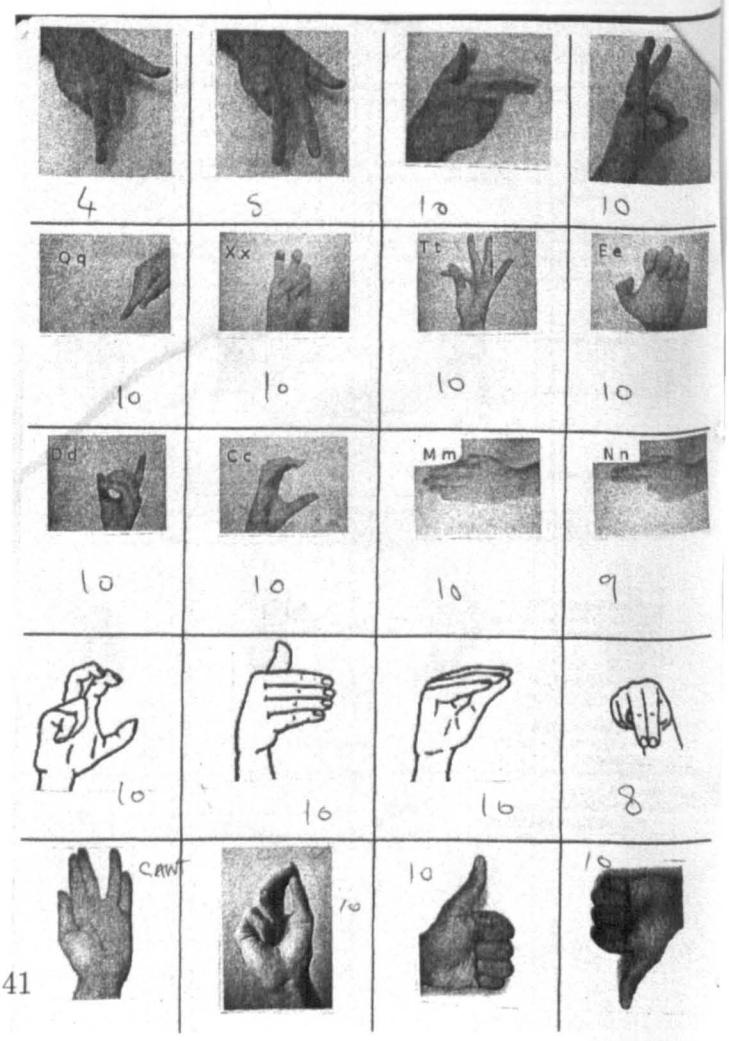



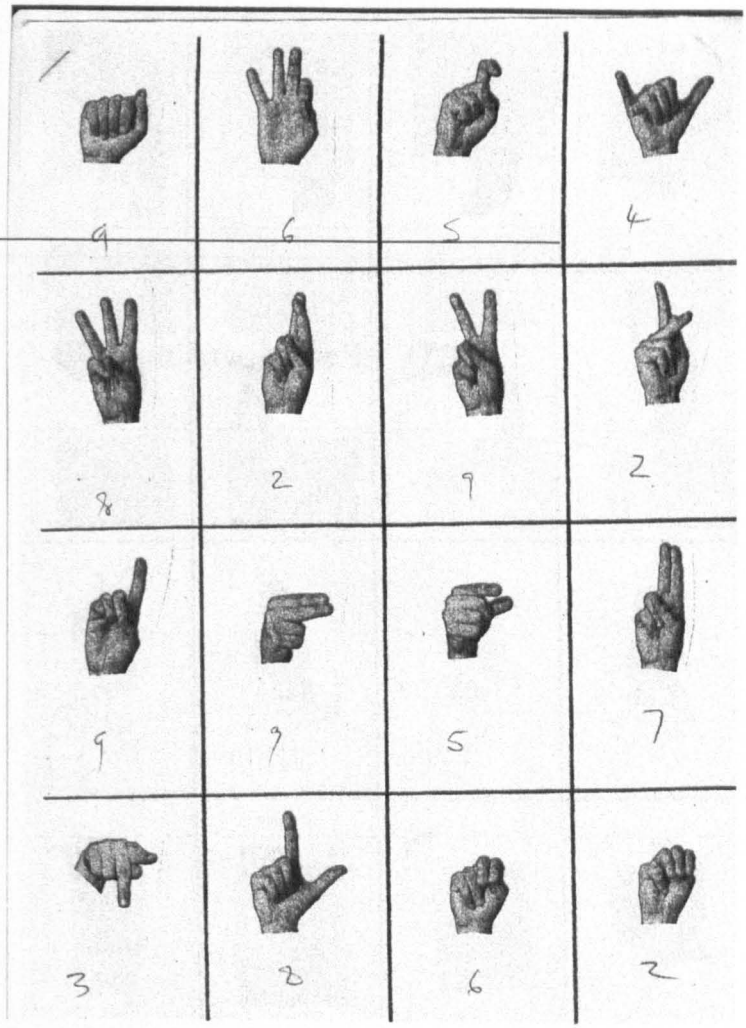
### .1.1.9 Participant ka2

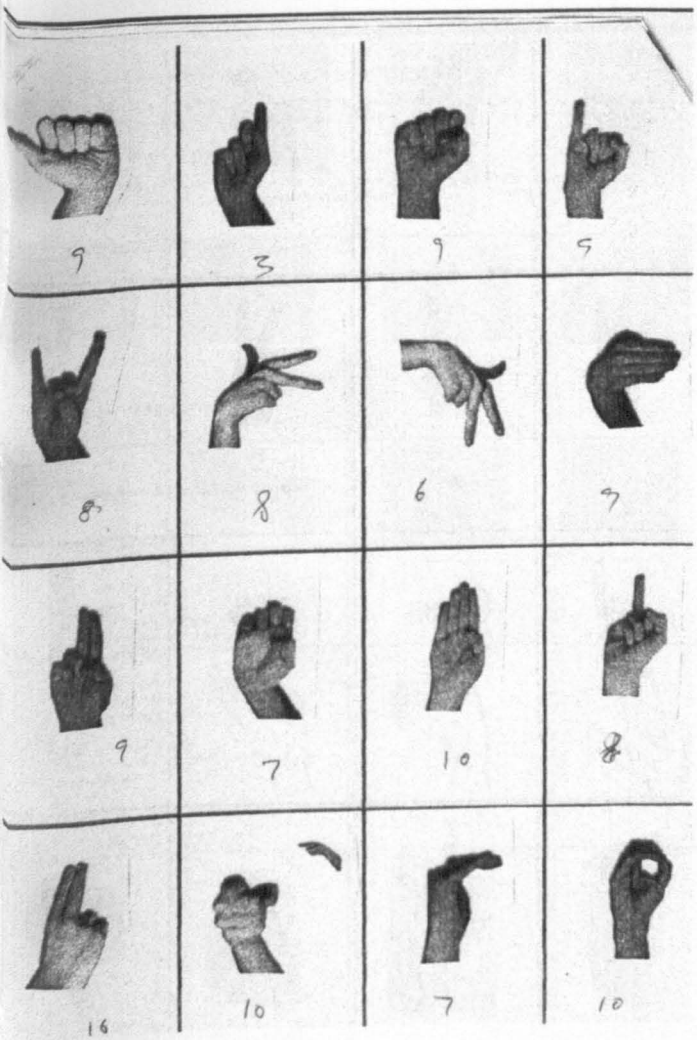


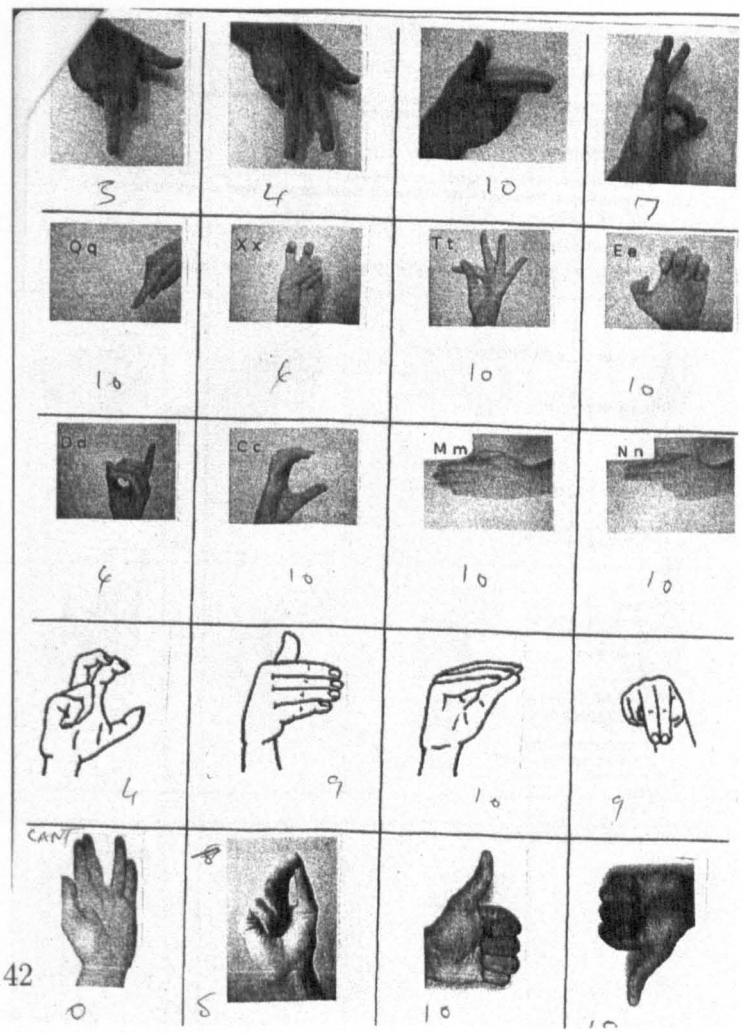



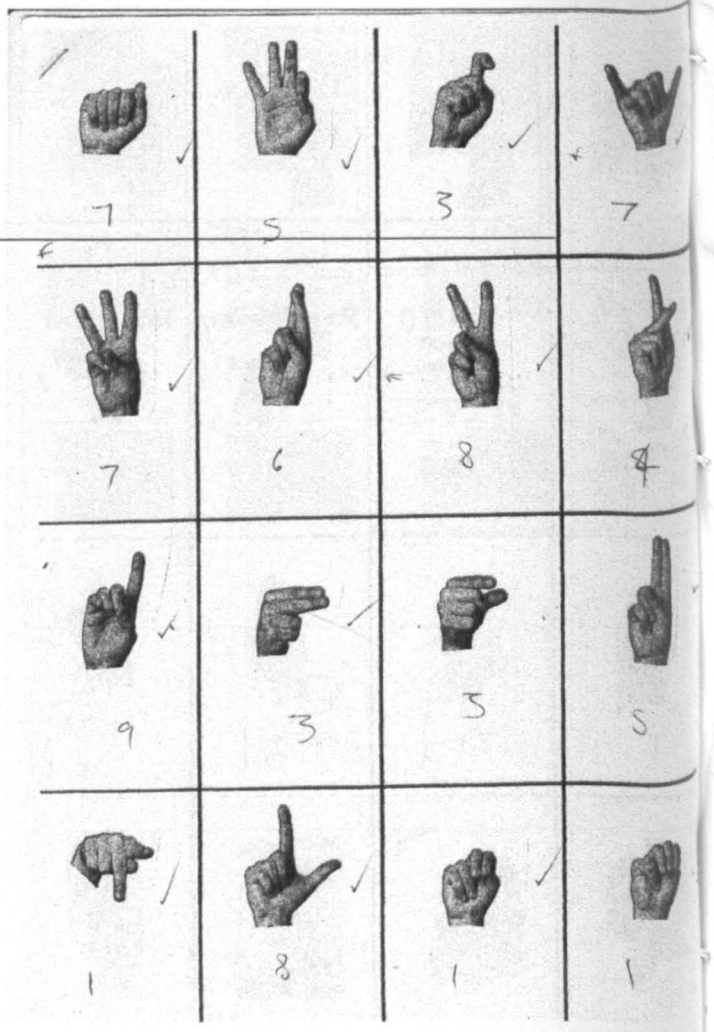
### .1.1.10 Participant la2

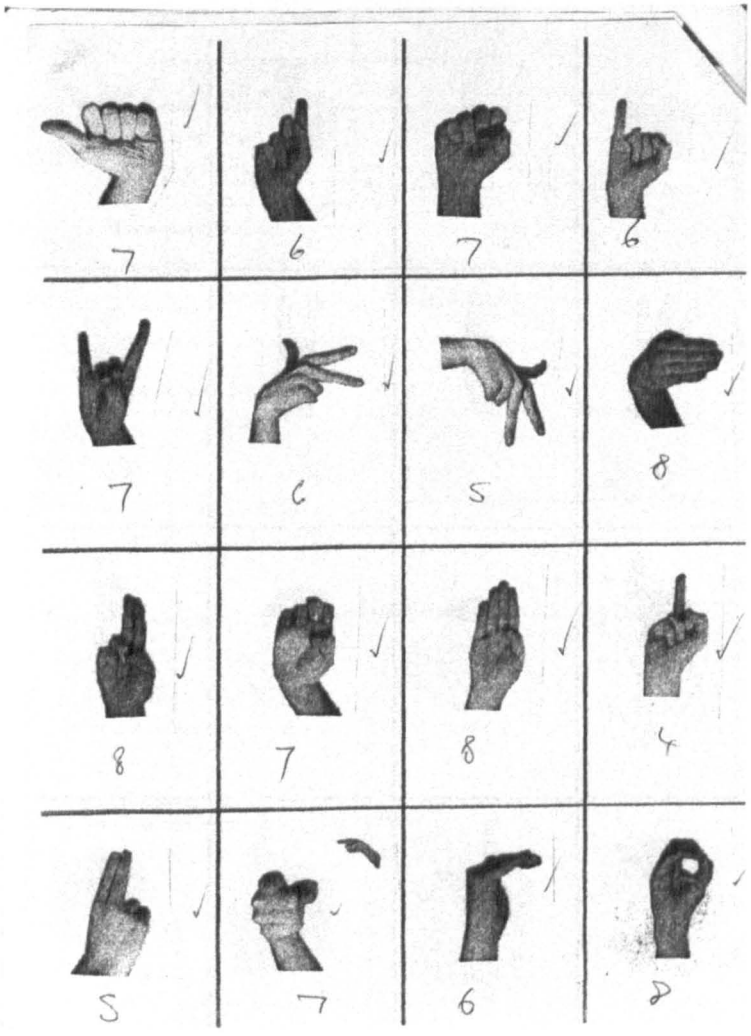


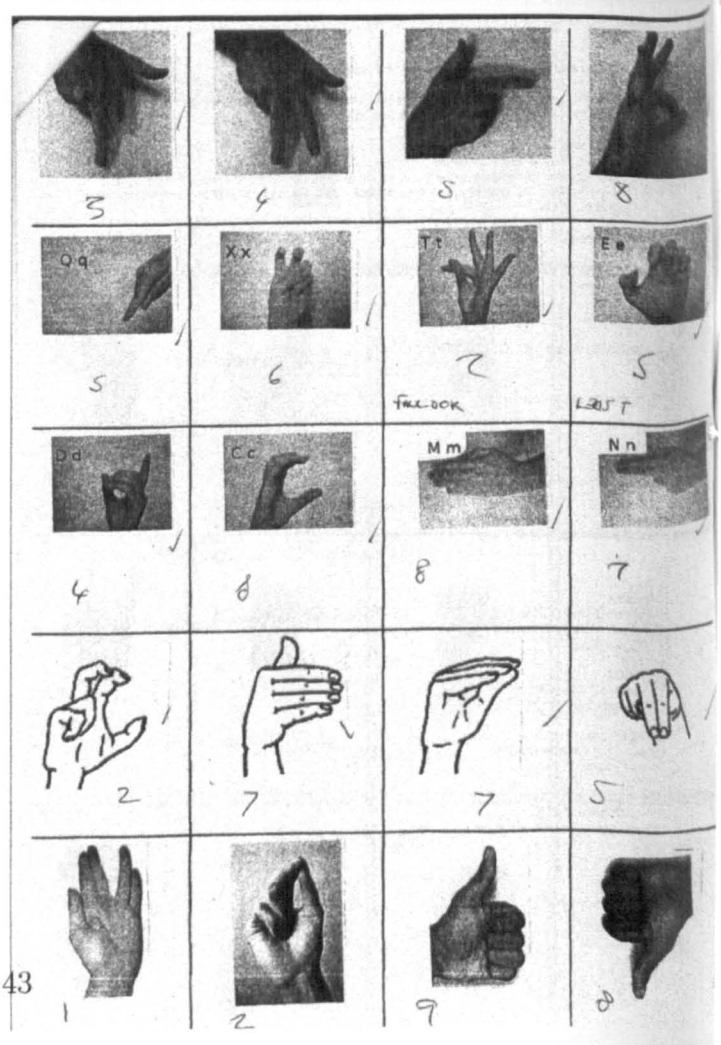



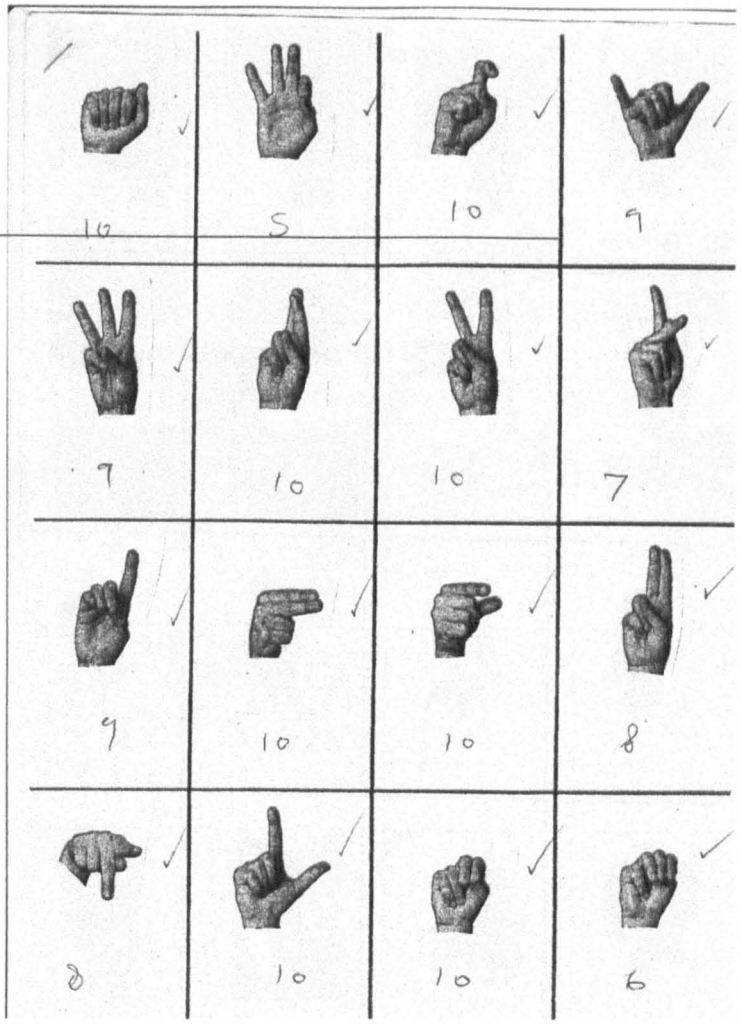
### .1.1.11 Participant ma1

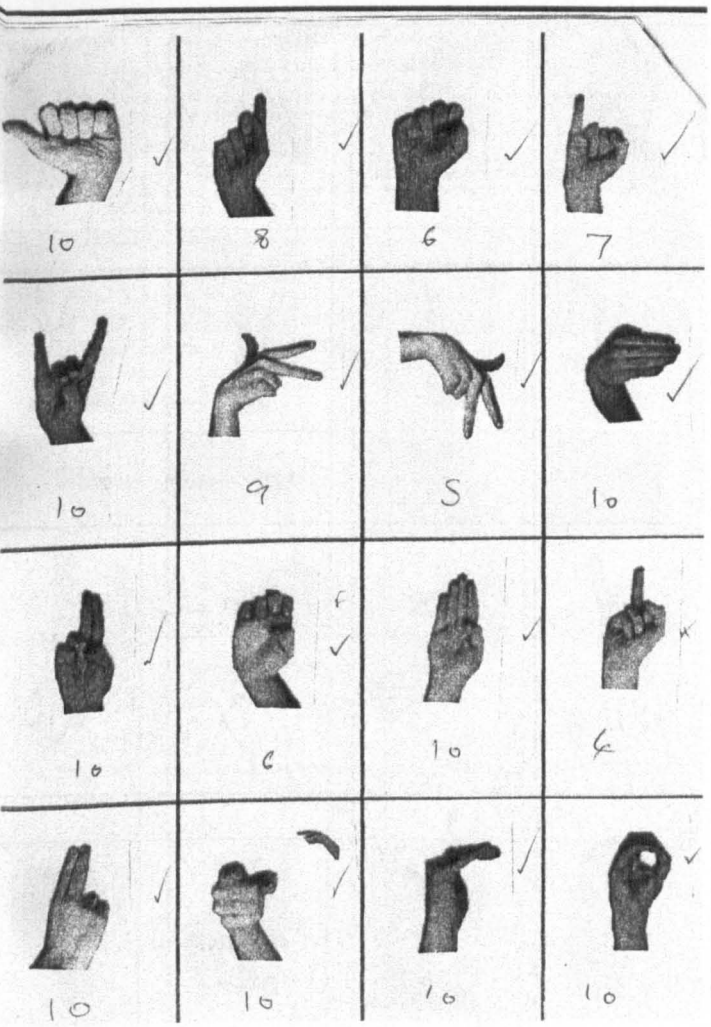


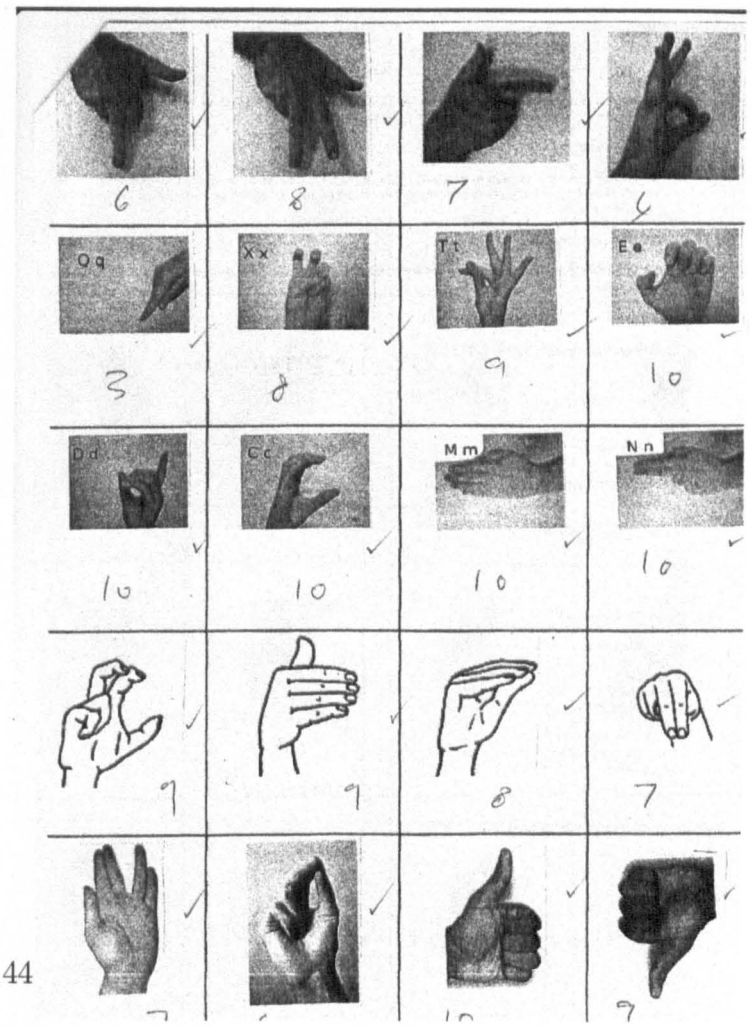




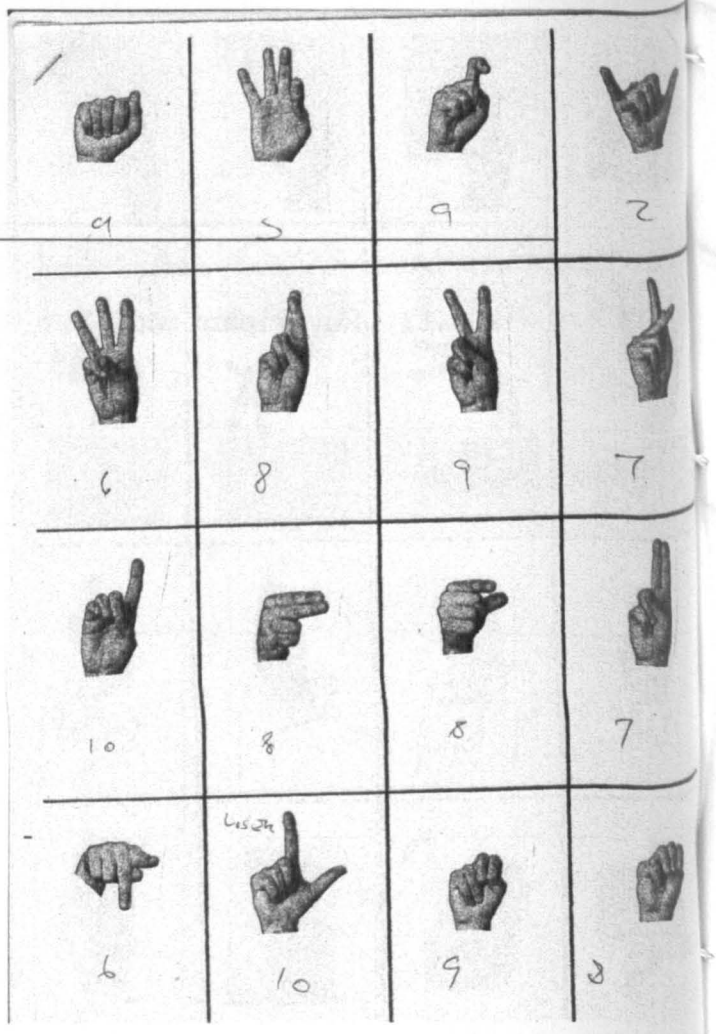
### .1.1.12 Participant mo2

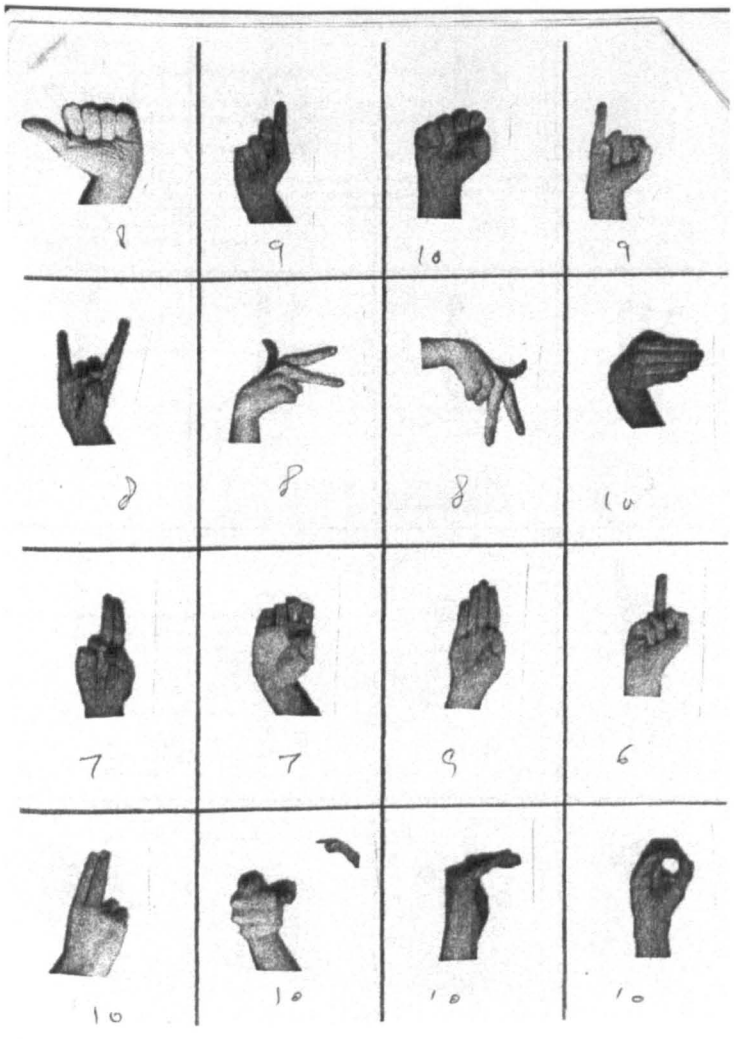


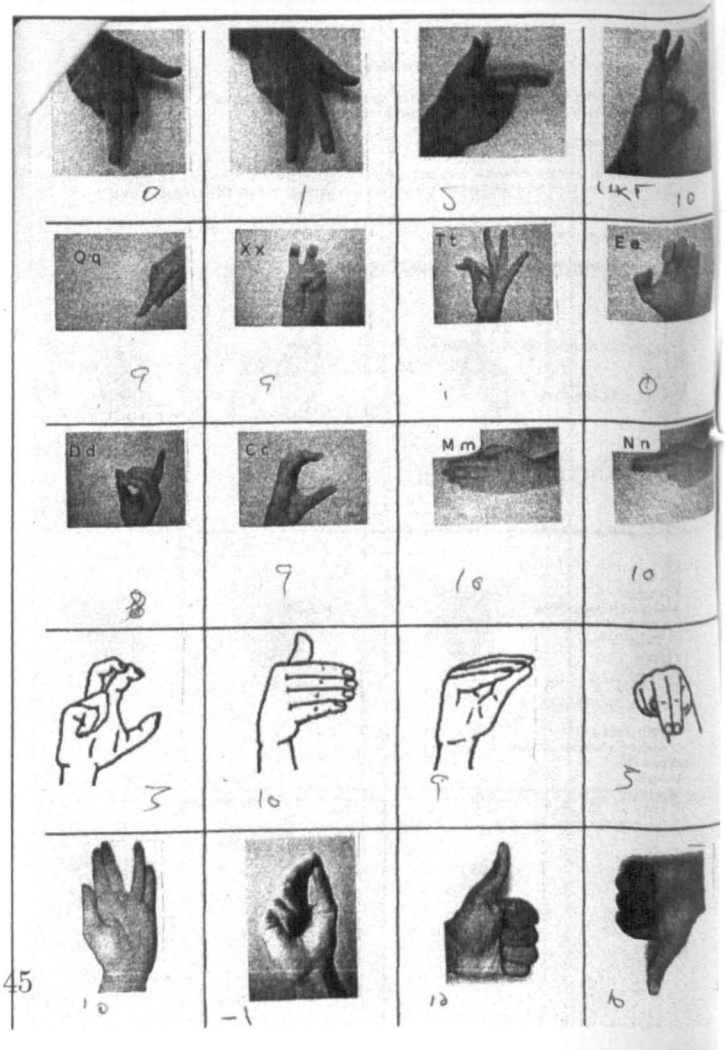




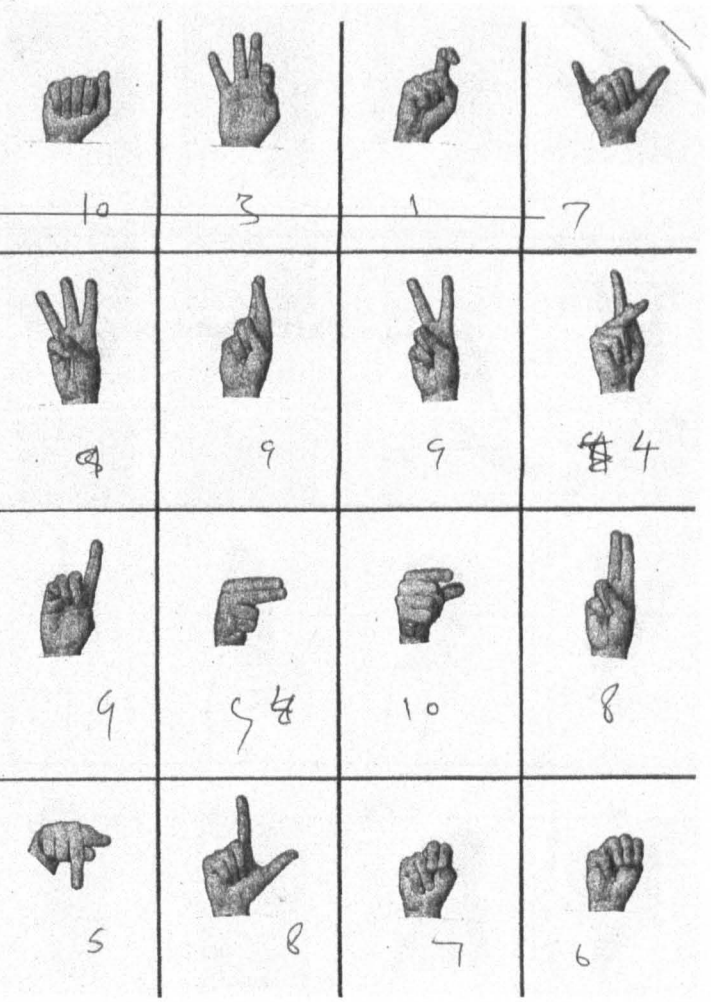
### .1.1.13 Participant my2

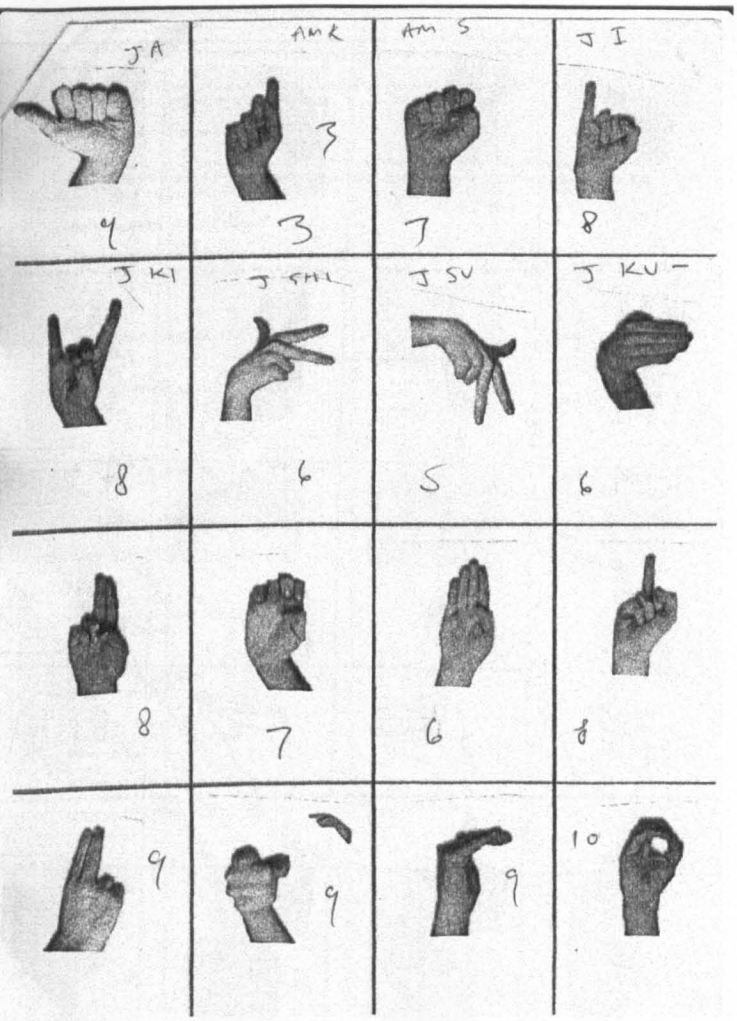


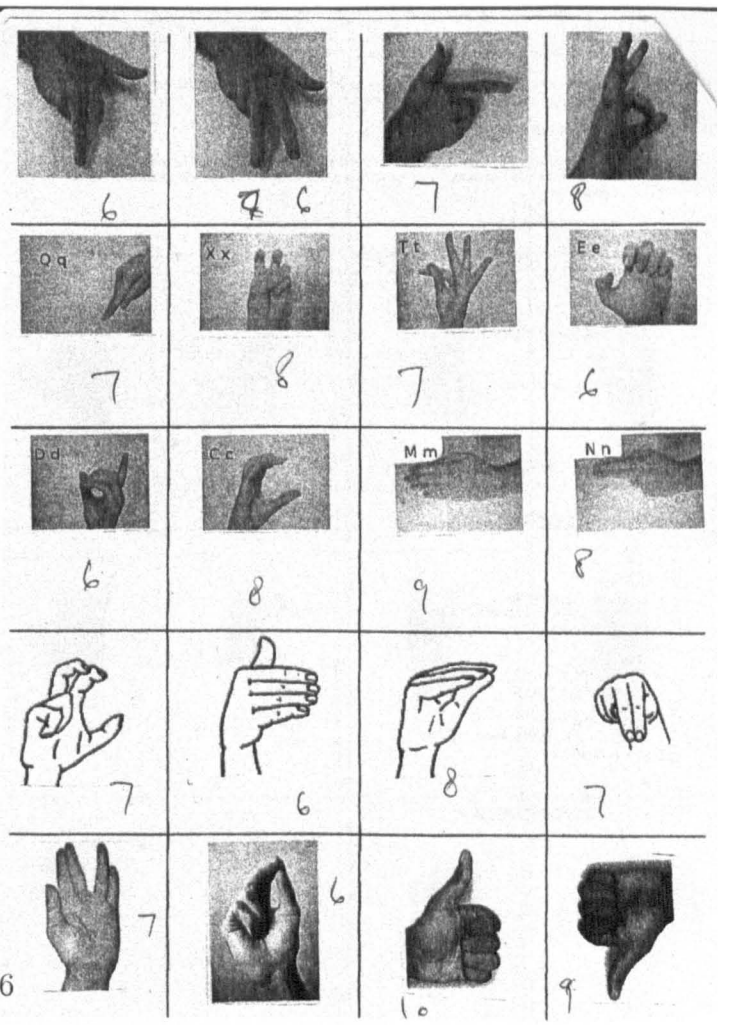




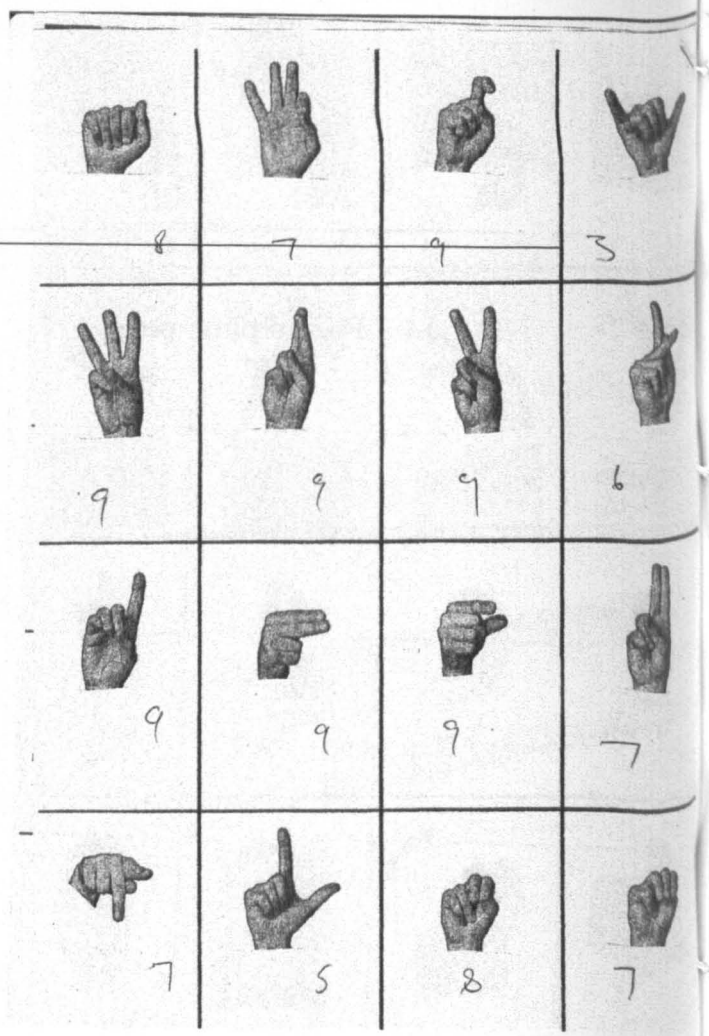
### .1.1.14 Participant pa1

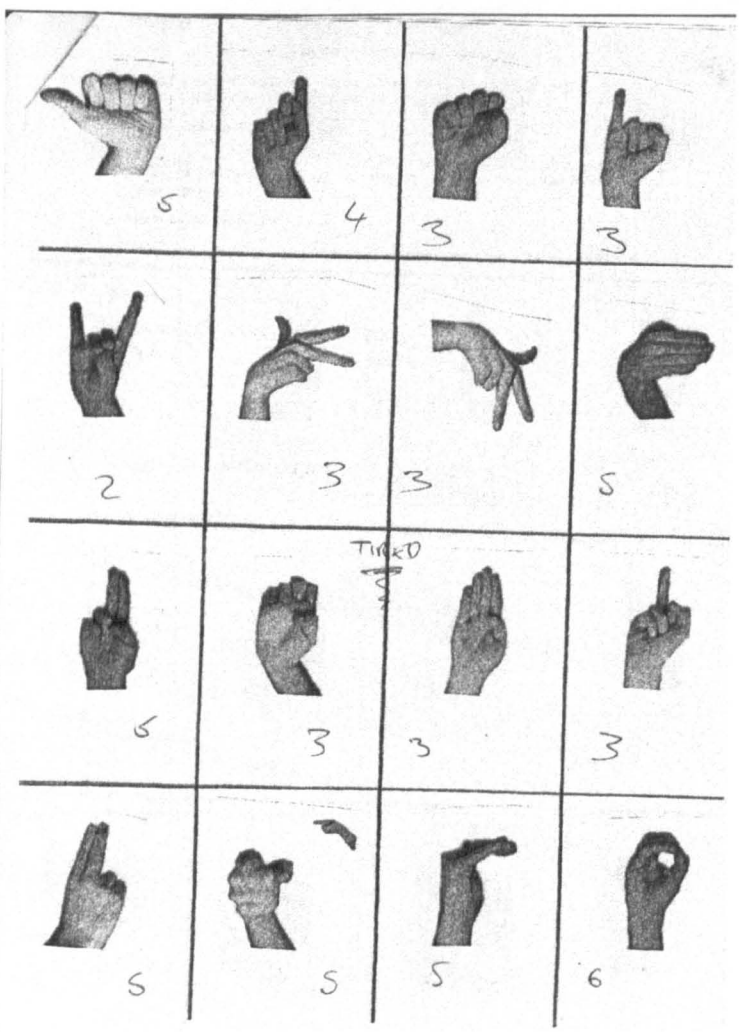


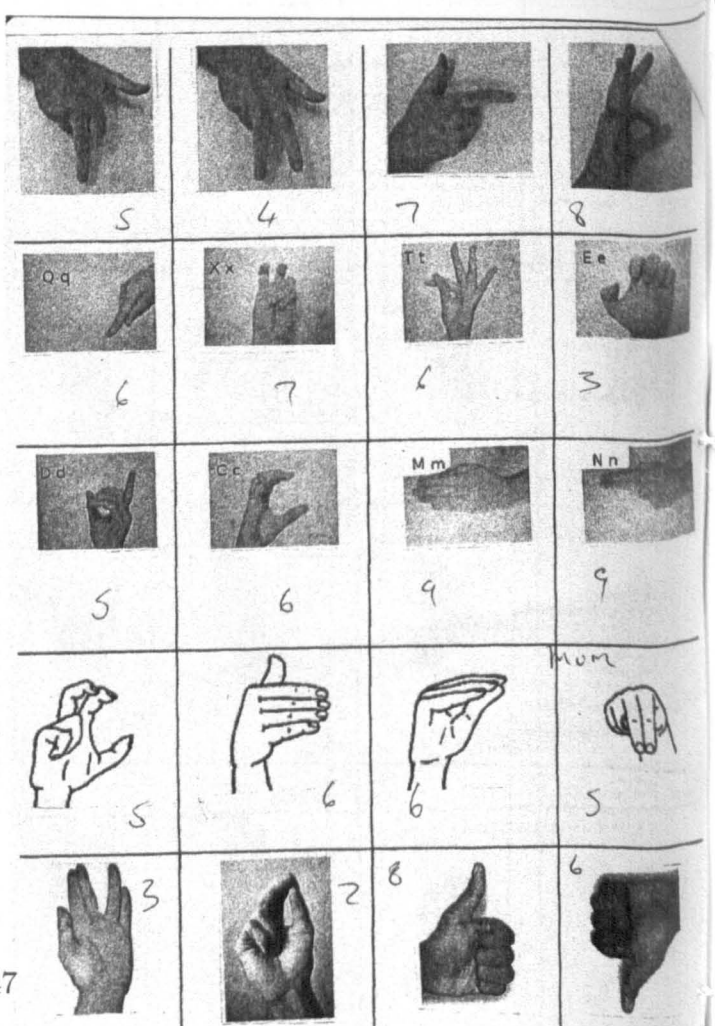


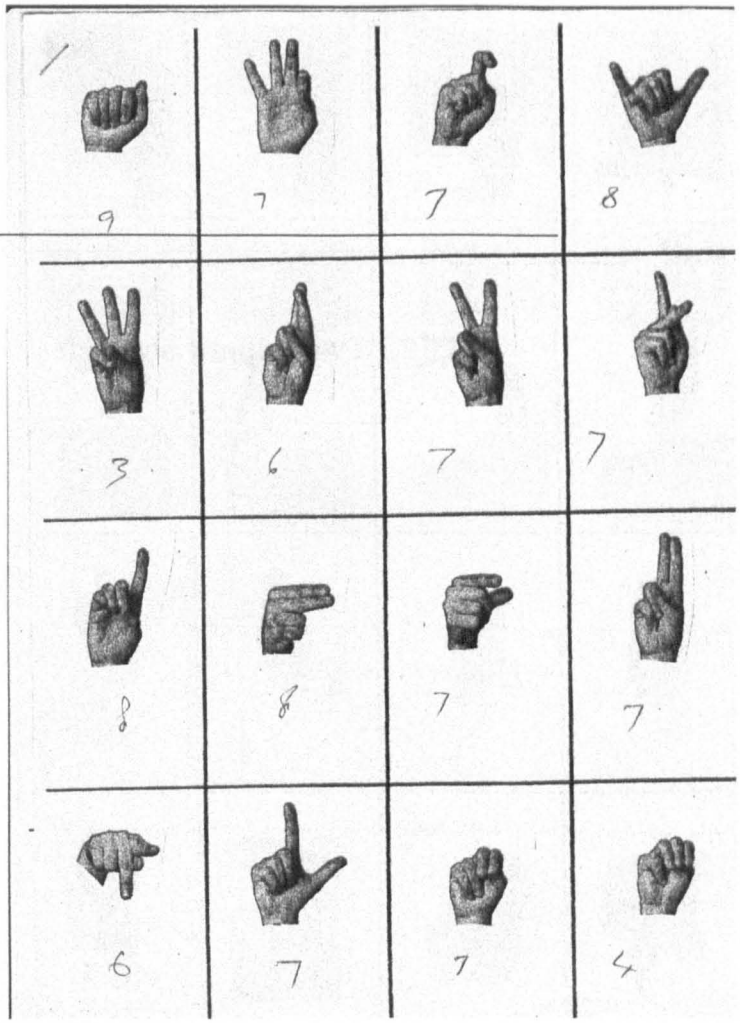
### .1.1.15 Participant ro2

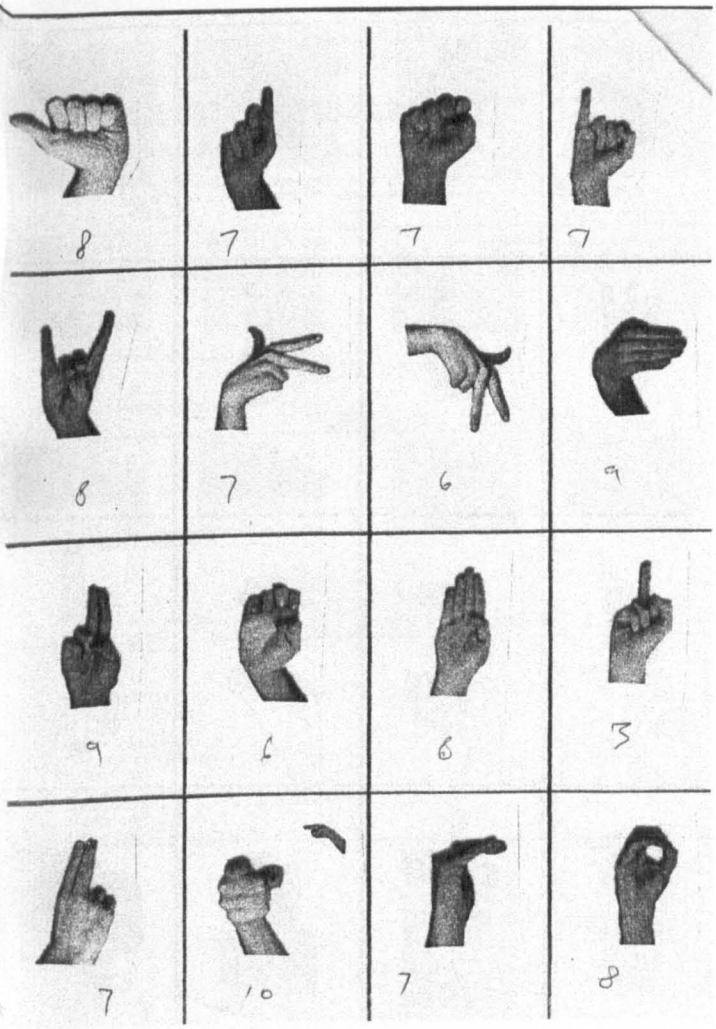


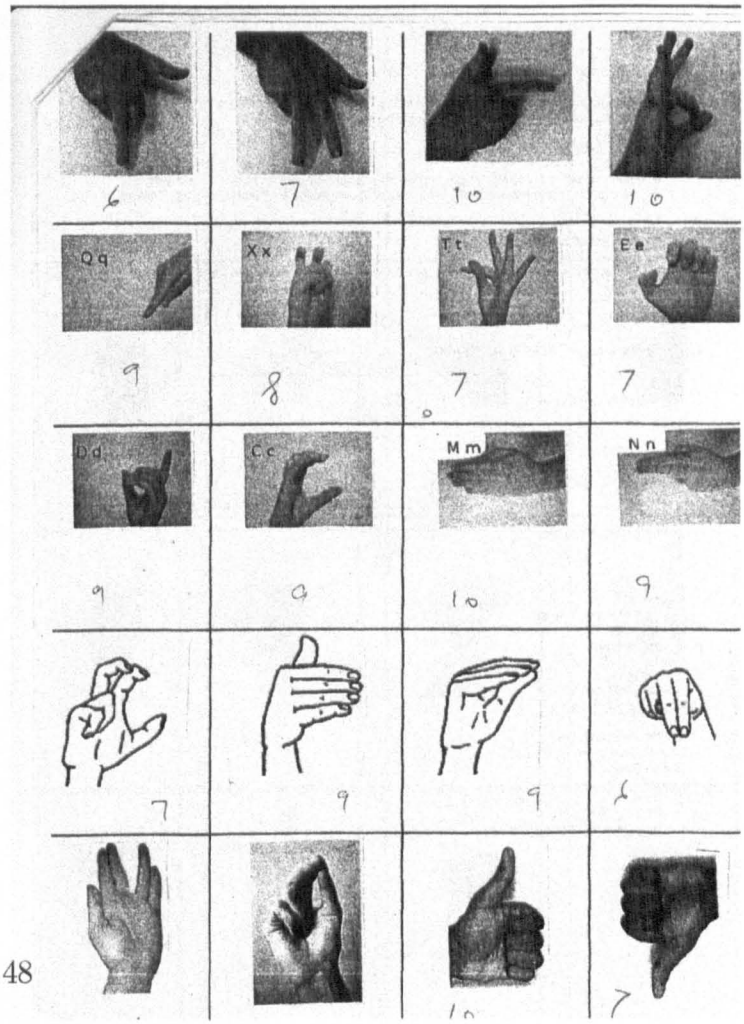



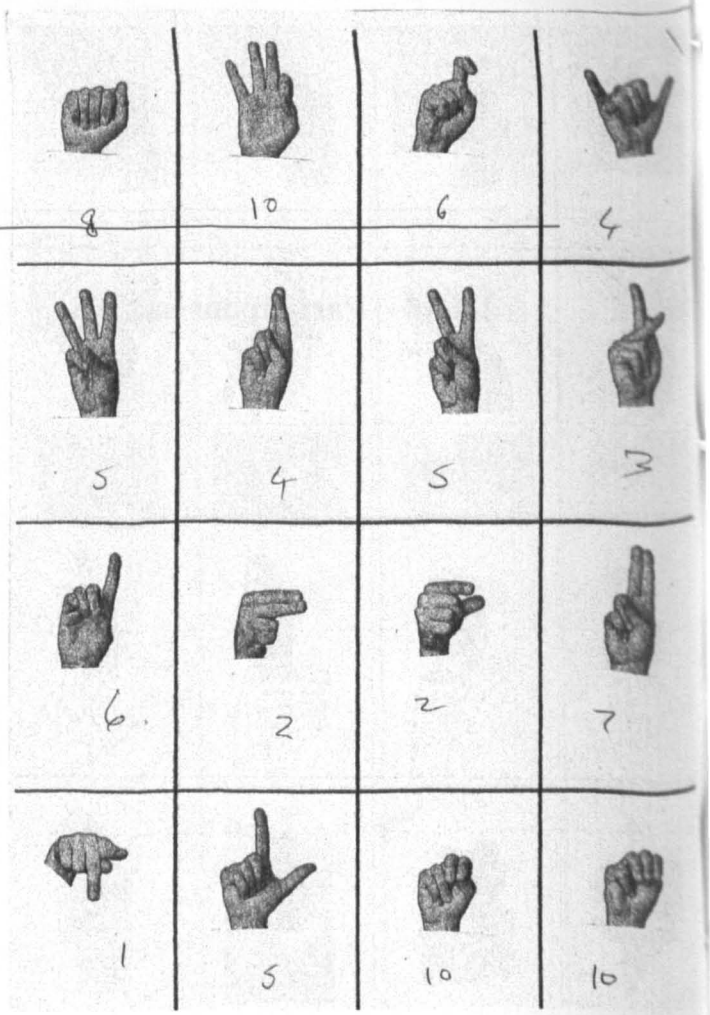
### .1.1.16 Participant sa2

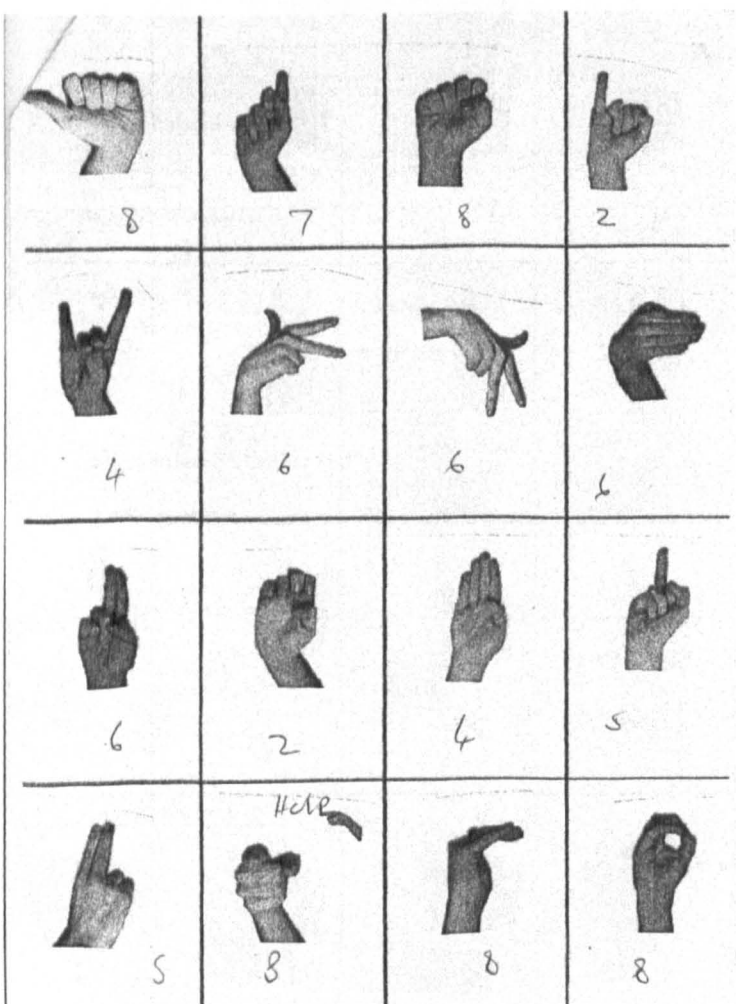


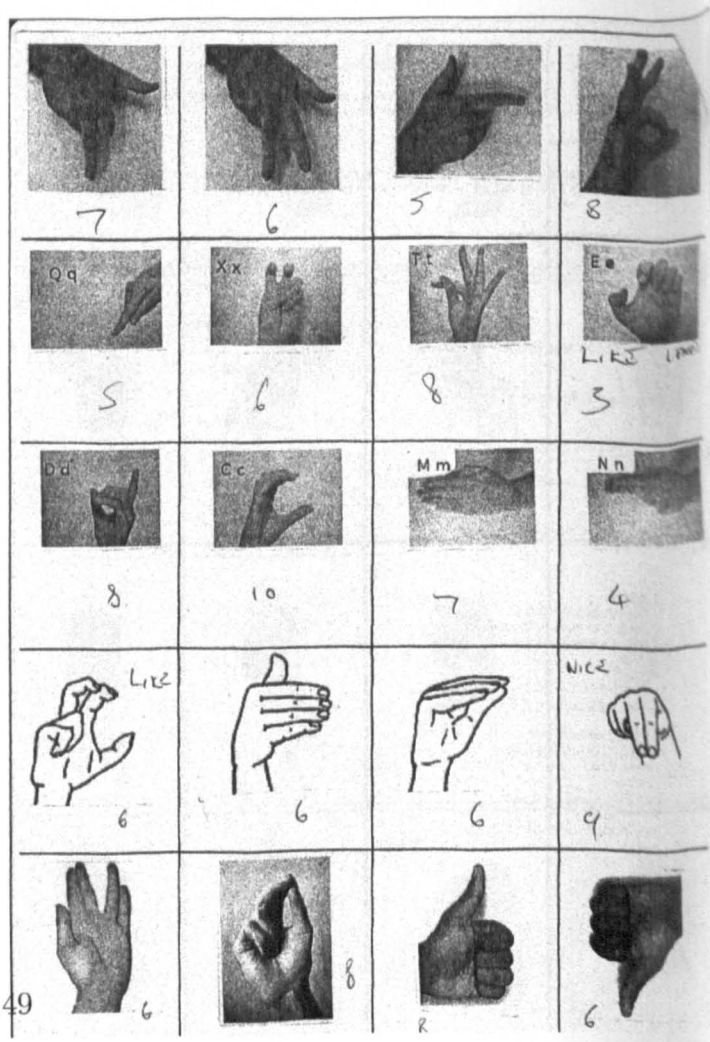



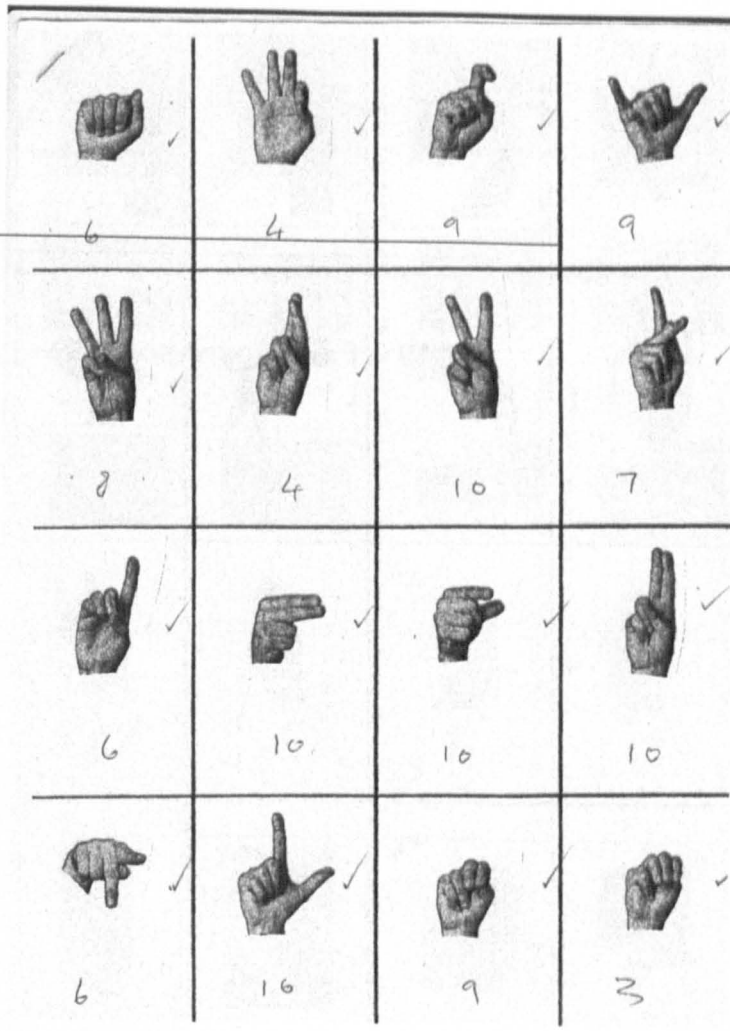
### .1.1.17 Participant sc2

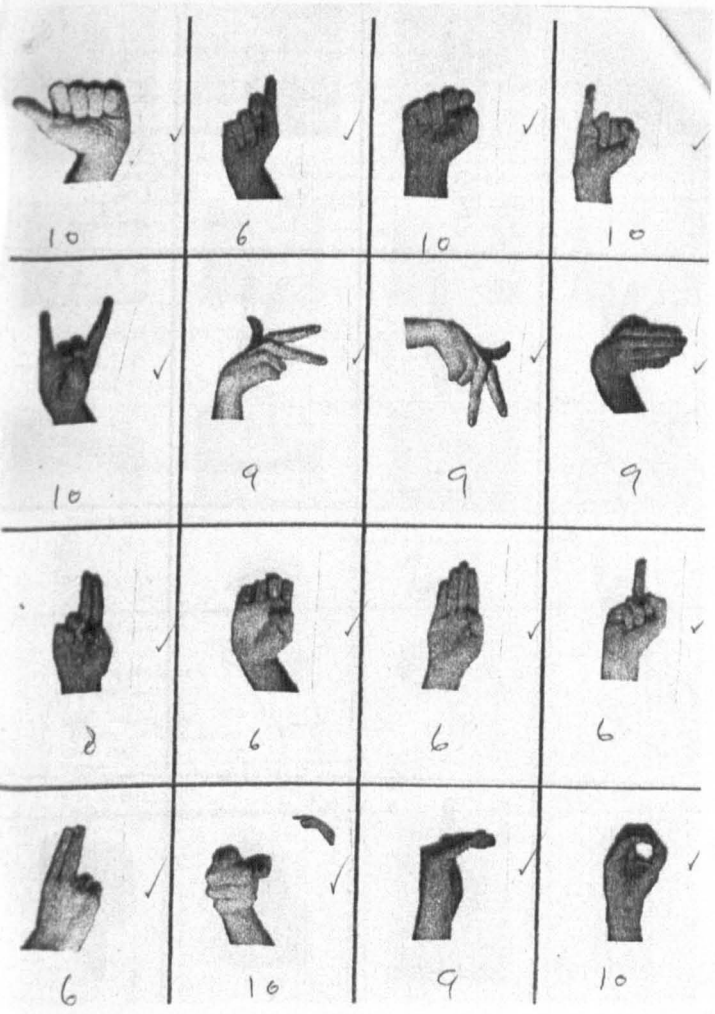


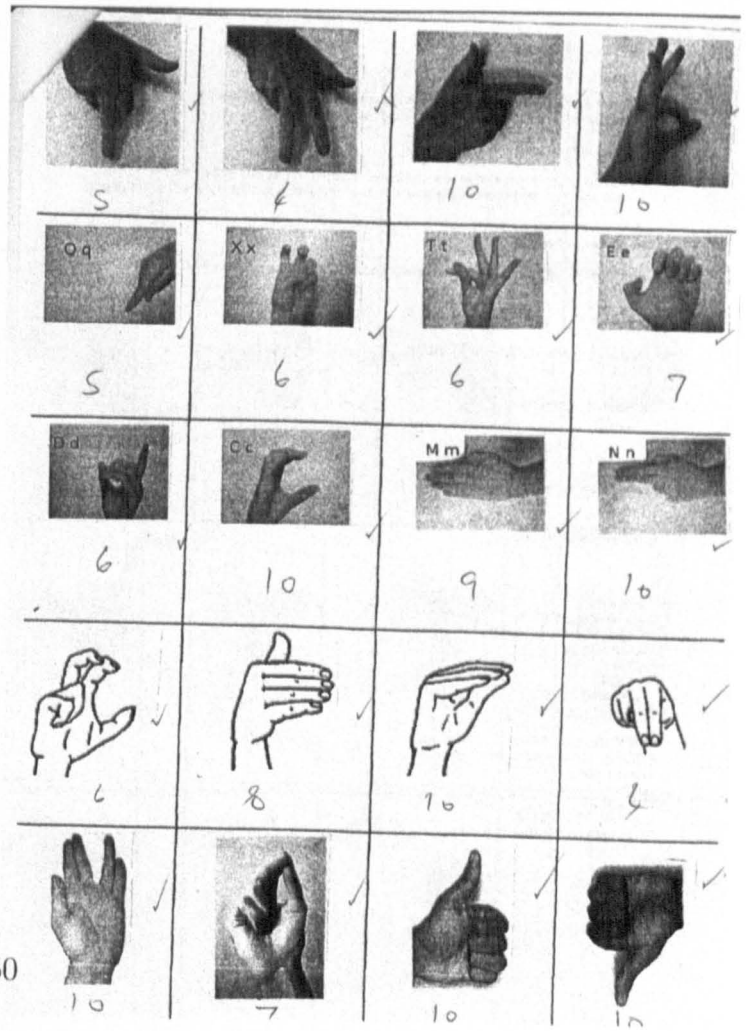



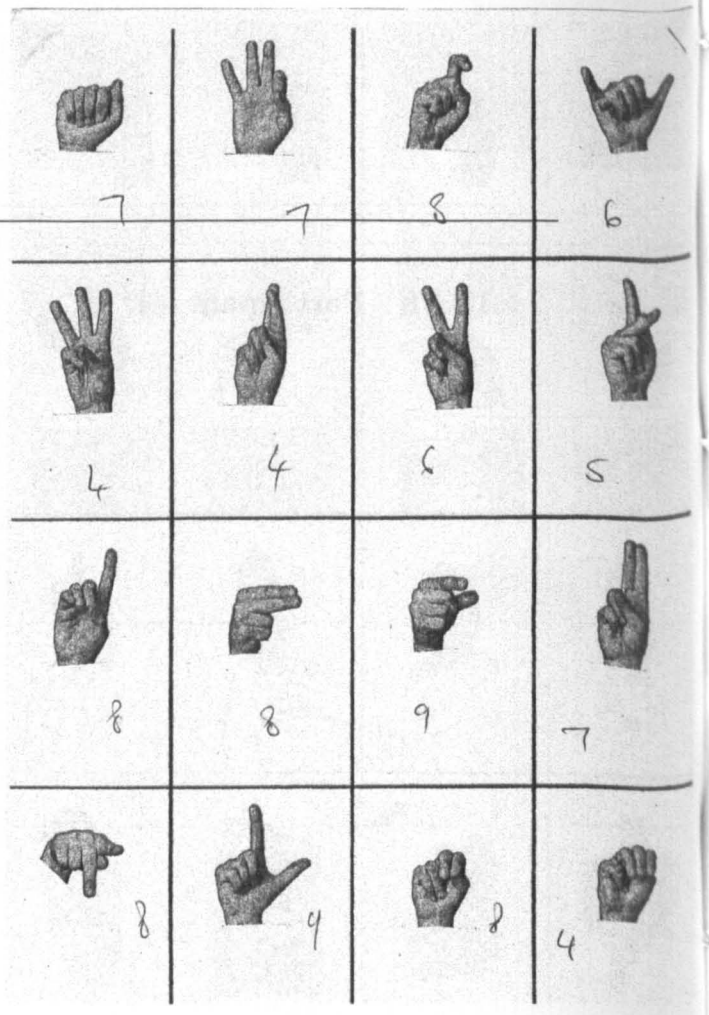
### .1.1.18 Participant si1

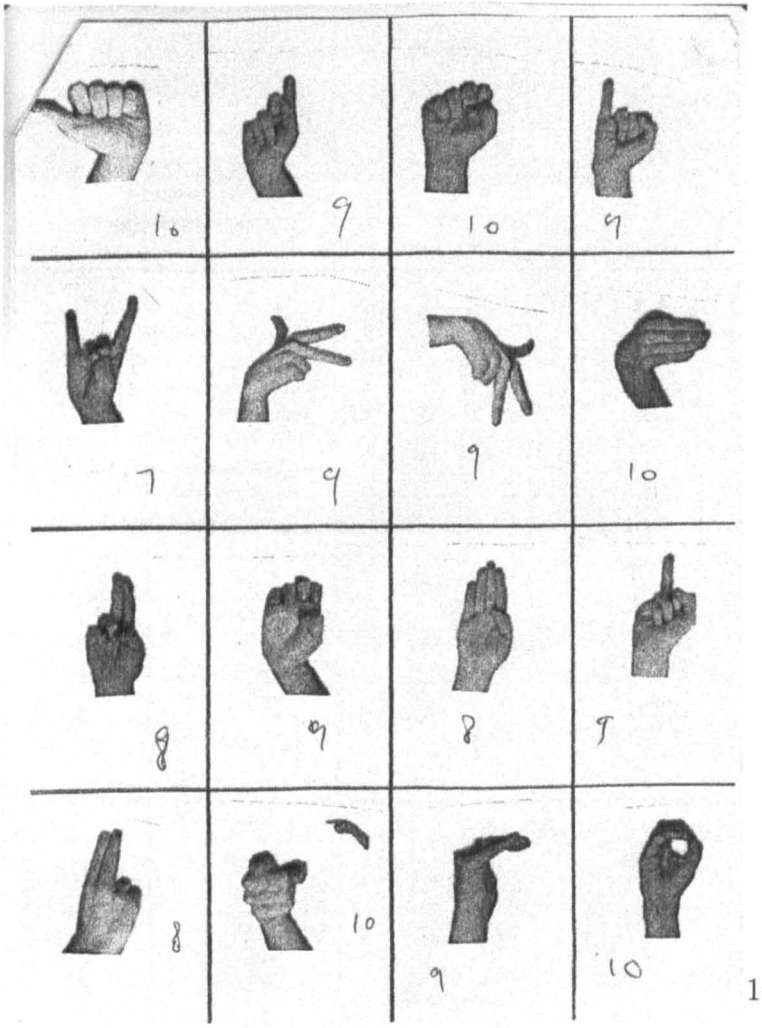


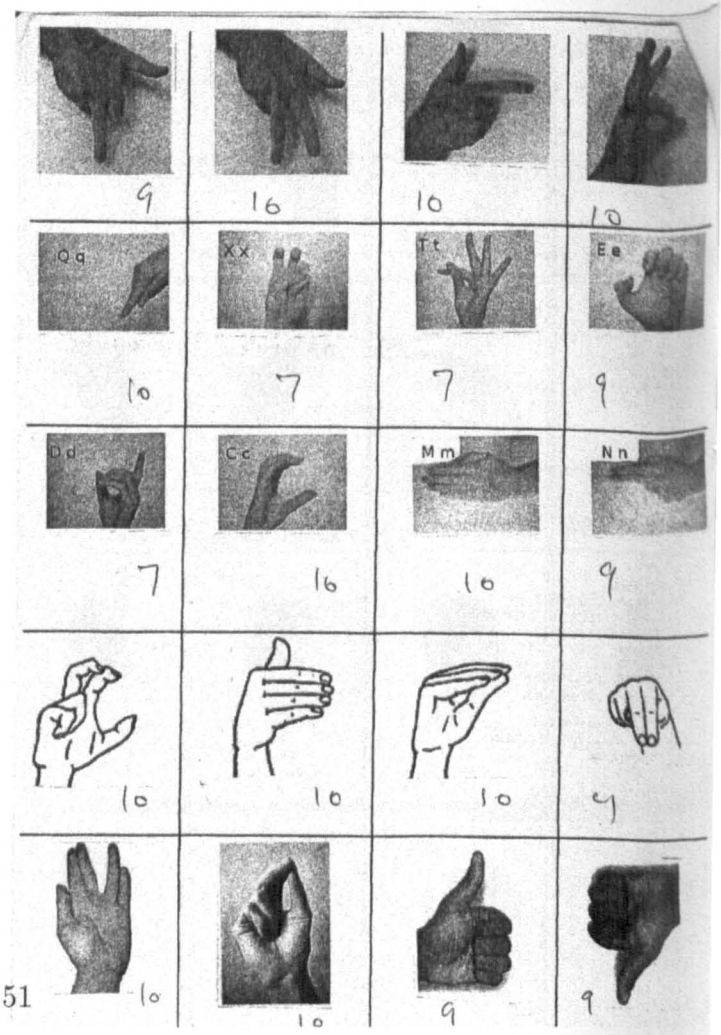



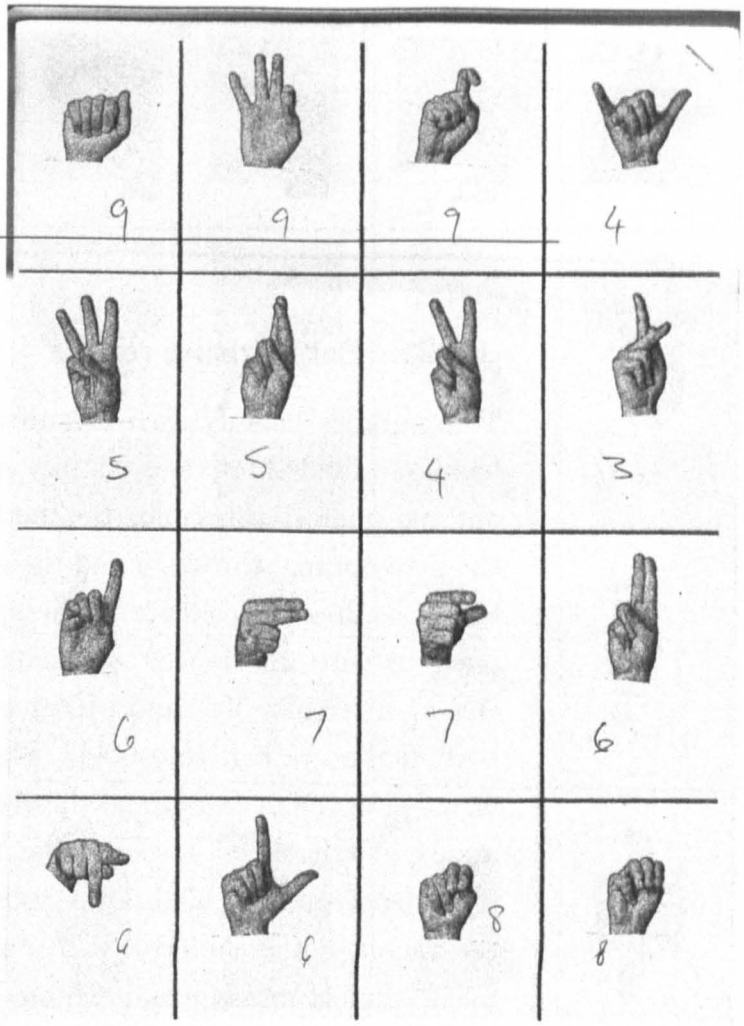
### .1.1.19 Participant so2

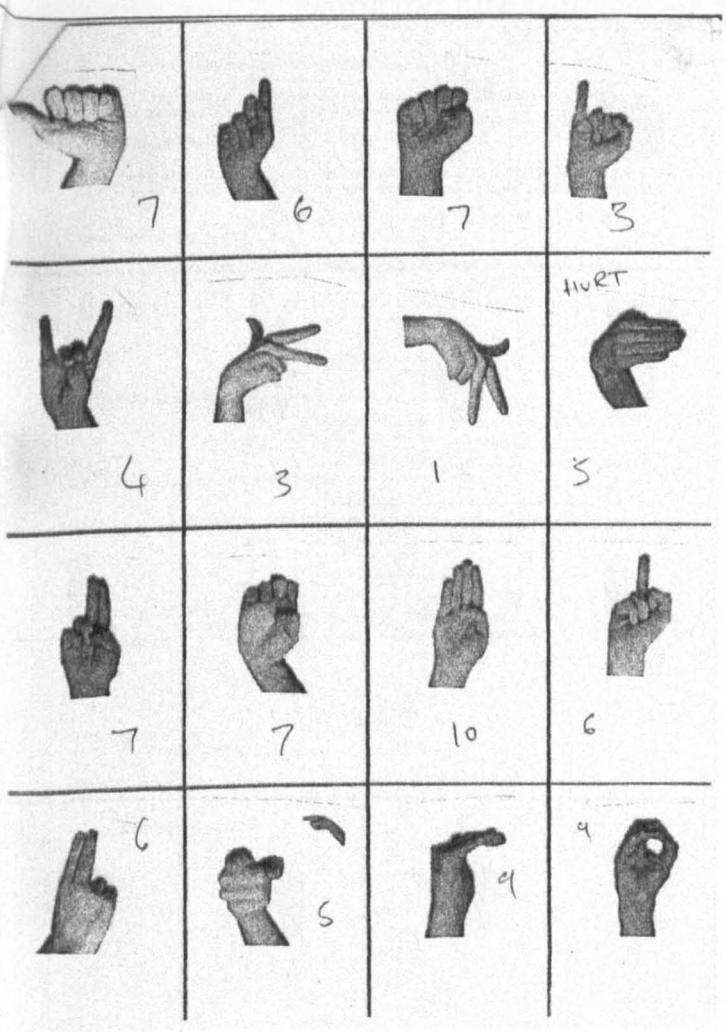


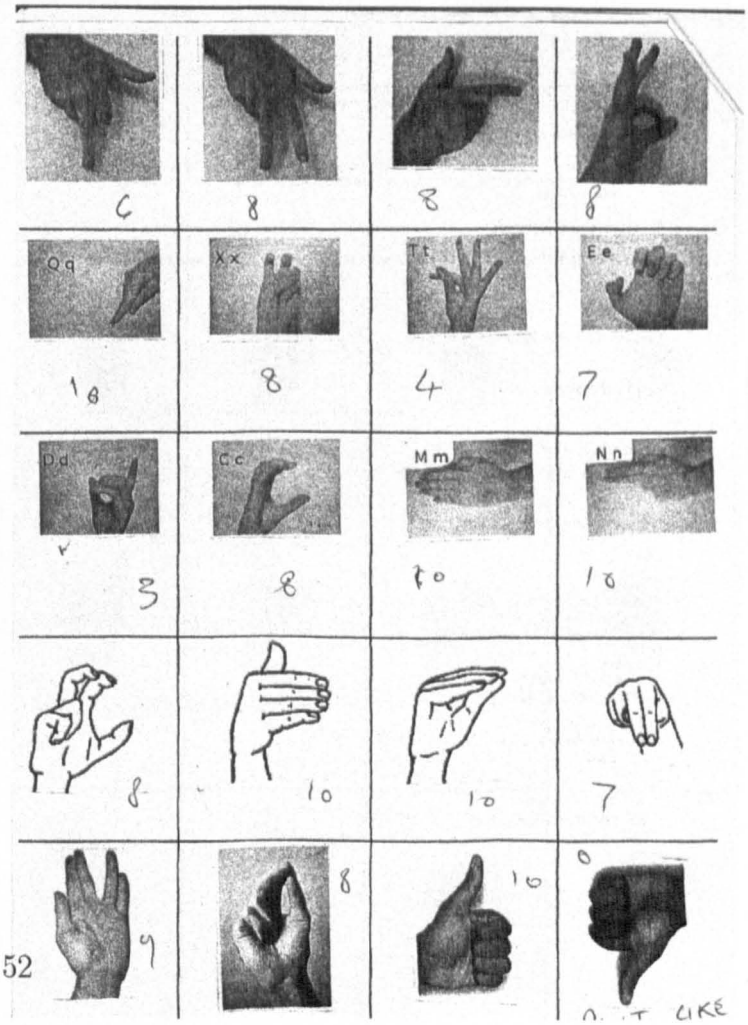



### .1.1.20 Participant to1



### .1.1.21 Categorising results

To accurately classify each posture the perceived physical exertion (ppe) index has been divided into five distinct subsets. These subsets are referred to as the optimal-comfort threshold; the meta-comfort band; the neo-optimal threshold; the beta-optimal threshold and the sub-optimal threshold. Shaded circles denote hand postures that adhere to the optimal-comfort threshold. Triangles identify postures with means that are marginally outside the optimal mean threshold. Blank circles identify hand postures that have a standard deviation fractionally beyond the optimal threshold. Blank circles and triangles represent postures included in the meta-optimal threshold. The hand shapes marked with a shaded square identify hand postures that fall within the beta-optimal threshold. The shaded polygons highlight two alternative representation of the same posture, the Japanese manual letter U. Notably, these two postures received an identical mean comfort measure and exhibits similarly high degrees of consensus amongst participants assessments. The proximity of these two assessments can be seen in figure 3.3, the positions of these postures are indicated with two arrows.

Table 4: Calculated perceived physical exertion (PPE) Mean and Standard deviation

| no. | Hand dact | PPE Score | Female PPE | Male PPE | PPE STD | Female STD | Male STD |
|---|---|---|---|---|---|---|---|
| 1 | American A | 7.2105 | 8.3636 | 5.625 | 2.6157 | 2.0136 | 2.6152 |
| 2 | American F | 6.5263 | 6.9091 | 6 | 1.8964 | 1.6404 | 2.2039 |
| 3 | American X | 7.0526 | 8 | 5.75 | 2.5706 | 1.6125 | 3.151 |
| 4 | American Y | 6.1053 | 5.9091 | 6.375 | 2.5797 | 2.8091 | 2.3867 |
| 5 | American W | 6.4211 | 6.4545 | 6.375 | 2.1426 | 2.3394 | 1.9955 |
| 6 | American R | 5.6842 | 6 | 5.25 | 2.4279 | 2.6077 | 2.252 |
| 7 | American V | 7.5789 | 7.7273 | 7.375 | 1.8048 | 1.4894 | 2.2638 |
| 8 | American K | 4.3158 | 4.9091 | 3.5 | 2.4279 | 2.6251 | 2 |
| 9 | American D | 8 | 8.1818 | 7.75 | 1.8257 | 1.6011 | 2.1876 |
| 10 | American H | 7.2105 | 7.4545 | 6.875 | 2.5944 | 2.6216 | 2.6959 |
| 11 | American G | 7.1579 | 7.2727 | 7 | 2.6929 | 3.003 | 2.3905 |
| 12 | American U | 7.3158 | 7.4545 | 7.125 | 1.7337 | 1.5725 | 2.031 |
| 13 | American P | 5.4211 | 4.8182 | 6.25 | 2.6101 | 2.6389 | 2.4928 |
| 14 | American L | 7.5263 | 7.5455 | 7.5 | 2.1952 | 2.3817 | 2.0702 |
| 15 | American N | 6.6316 | 7.6364 | 5.25 | 2.4315 | 1.6293 | 2.7646 |
| 16 | American M | 4.6842 | 5.7273 | 3.25 | 2.6045 | 2.149 | 2.6049 |
| 17 | Japanese A | 8.1579 | 7.9091 | 8.5 | 1.7405 | 2.0715 | 1.1952 |
| 18 | American K2 | 6.3158 | 6.6364 | 5.875 | 1.8273 | 1.9117 | 1.7269 |
| 19 | American S | 7.4737 | 7.5455 | 7.375 | 1.9542 | 2.1616 | 1.7678 |
| 20 | Japanese I | 6.3684 | 6.1818 | 6.625 | 2.5213 | 2.6007 | 2.56 |
| 21 | Japanese KI | 6.8421 | 6.8182 | 6.875 | 2.4327 | 2.892 | 1.8077 |
| 22 | Japanese SHI | 6.5789 | 7 | 6 | 1.9809 | 2.1448 | 1.6903 |
| 23 | Japanese SU | 5.8947 | 5.8182 | 6 | 2.2827 | 1.8878 | 2.8785 |
| 24 | Japanese KU | 8 | 8.2727 | 7.625 | 1.8257 | 1.9022 | 1.7678 |
| 25 | Japanese U | 7.3158 | 7.3636 | 7.25 | 1.5653 | 1.7477 | 1.3887 |
| 26 | Japanese E | 5.8947 | 5.9091 | 5.875 | 1.8825 | 2.3856 | 0.99103 |

Table 5: Calculated perceived physical exertion (PPE) Mean and Standard deviation cont....

| no. | Hand dact | PPE Score | Female PPE | Male PPE | PPE STD | Female STD | Male STD |
|---|---|---|---|---|---|---|---|
| 27 | Japanese KE | 7.3684 | 7 | 7.875 | 2.5649 | 2.9665 | 1.9594 |
| 28 | Japanese SE | 6.1053 | 5.1818 | 7.375 | 2.5143 | 2.4008 | 2.1998 |
| 29 | Japanese TO | 7.4737 | 7.3636 | 7.625 | 1.9824 | 2.2033 | 1.7678 |
| 30 | Japanese SO | 8.7368 | 9.0909 | 8.25 | 1.6614 | 1.5783 | 1.7525 |
| 31 | Japanese KO | 7.5263 | 7.9091 | 7 | 2.0102 | 1.7581 | 2.3299 |
| 32 | Japanese O | 8.6316 | 9.1818 | 7.875 | 2.0058 | 1.328 | 2.5877 |
| 33 | Korean K | 4.7895 | 4.6364 | 5 | 2.3471 | 2.6181 | 2.0702 |
| 34 | Korean J | 5.4737 | 5.1818 | 5.875 | 2.2698 | 2.6007 | 1.8077 |
| 35 | Korean N | 7.5789 | 7.7273 | 7.375 | 1.9809 | 2.1019 | 1.9226 |
| 36 | Korean O | 8.2632 | 8.2727 | 8.25 | 1.6945 | 1.8488 | 1.5811 |
| 37 | French Q | 6.8421 | 7.4545 | 6 | 2.2426 | 2.3394 | 1.9272 |
| 38 | French X | 6.8421 | 7.0909 | 6.5 | 1.8032 | 1.9212 | 1.6903 |
| 39 | French T | 5.9474 | 6.6364 | 5 | 2.527 | 2.6934 | 2.0702 |
| 40 | French E | 6.2632 | 6.8182 | 5.5 | 2.922 | 3.6556 | 1.3093 |
| 41 | French D | 6.6316 | 7.0909 | 6 | 2.3854 | 2.7732 | 1.6903 |
| 42 | French C | 8.4211 | 9.1818 | 7.375 | 1.8048 | 1.1677 | 2.0659 |
| 43 | French M | 9.0526 | 9.2727 | 8.75 | 1.2236 | 1.1909 | 1.2817 |
| 44 | French N | 8.7895 | 8.8182 | 8.75 | 1.5484 | 1.7215 | 1.3887 |
| 45 | Polish C | 6.0526 | 6.6364 | 5.25 | 2.2724 | 2.3779 | 1.9821 |
| 46 | Polish M | 7.9474 | 8.5455 | 7.125 | 1.8097 | 1.4397 | 2.031 |
| 47 | Polish E | 7.9474 | 8.0909 | 7.75 | 1.8401 | 1.8141 | 1.9821 |
| 48 | Irish N | 6.3158 | 6.7273 | 5.75 | 1.7014 | 1.954 | 1.165 |
| 49 | Salaam | 6.3158 | 6 | 6.75 | 3.4649 | 3.873 | 3.0119 |
| 50 | Kubera | 5.9474 | 6.1818 | 5.625 | 2.7983 | 3.4005 | 1.8468 |
| 51 | Japanese TA | 9.3158 | 9.1818 | 9.5 | 1.1082 | 1.328 | 0.75593 |
| 52 | Japanese TA r | 7.6842 | 7.6364 | 7.75 | 2.8295 | 2.5796 | 3.3274 |

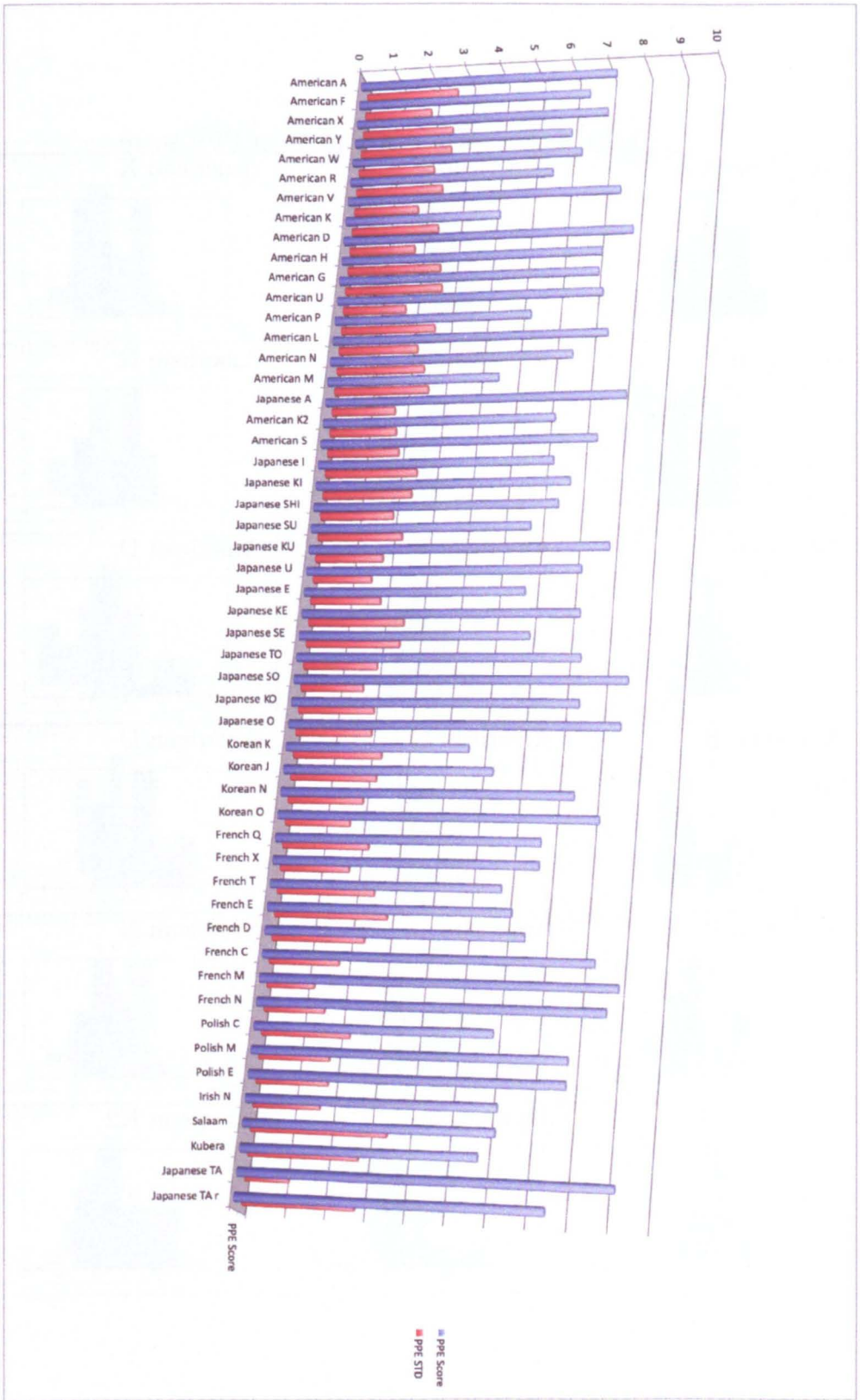



Figure 1: Illustrates threshold variance

Table 6: Gaussian distribution of PPE score

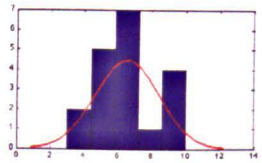
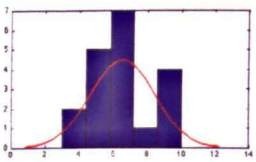
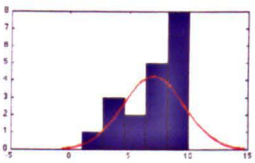
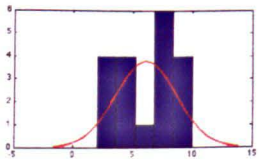
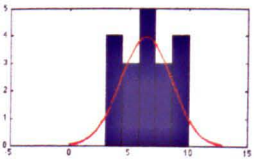
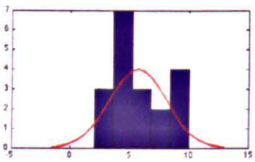
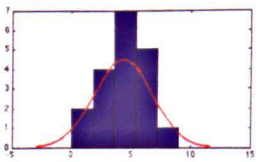
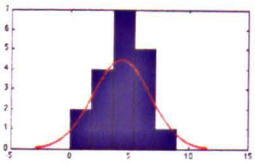
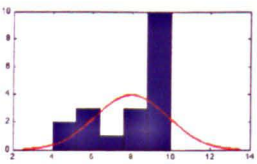
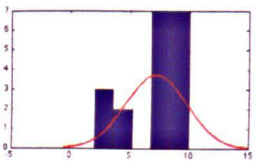
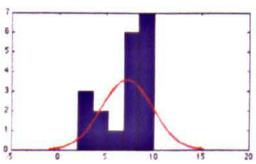
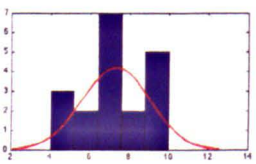
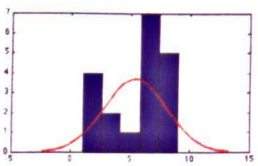
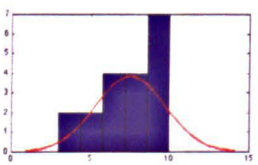
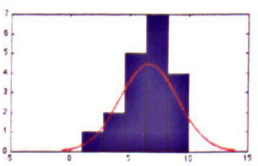
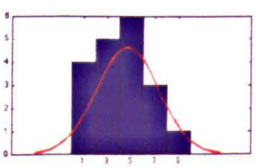
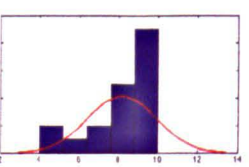
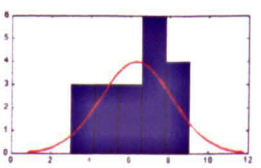
| | | |
|---|---|---|
| American A  | American F  | American X  |
| American Y  | American W  | American R  |
| American V  | American K  | American D  |
| American H  | American G  | American U  |
| American P  | American L  | American N  |
| American M  | Japanese A  | American K2  |

Table 7: Gaussian distribution of PPE score

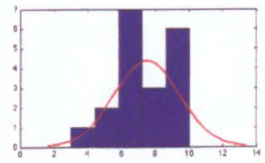
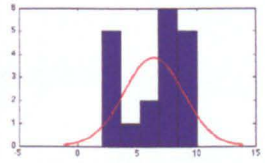
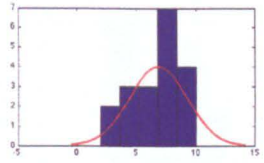
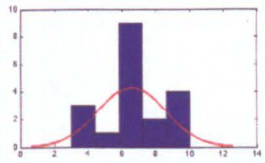
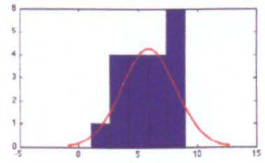
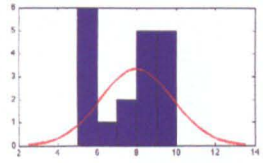
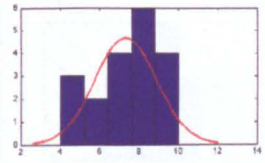
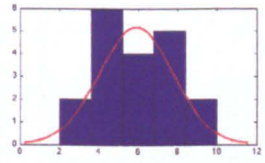
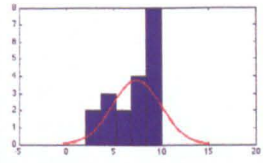
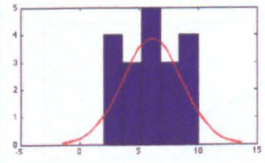
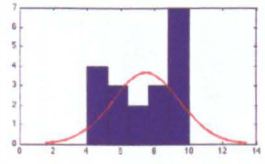
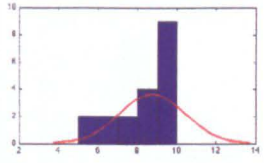
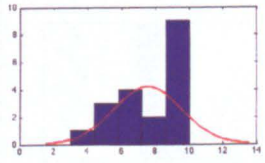
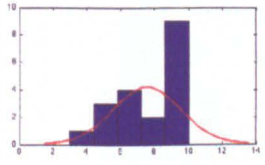
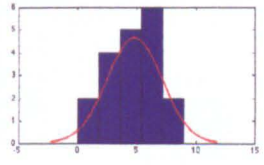
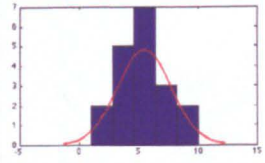
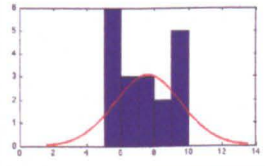
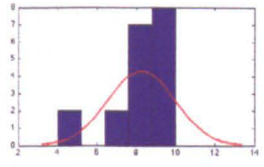
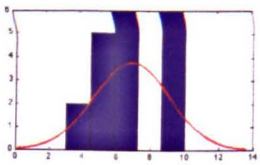
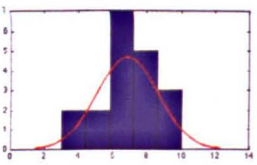
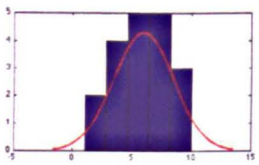
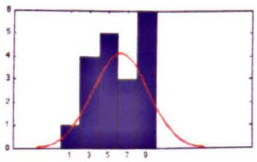
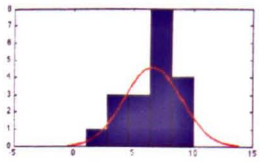
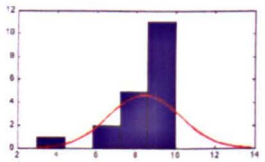
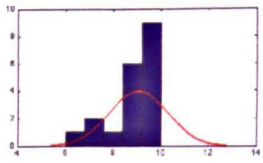
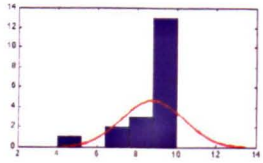
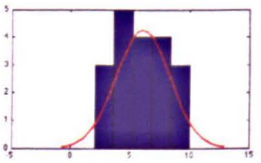
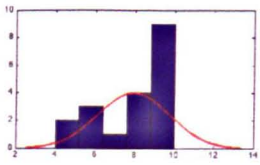
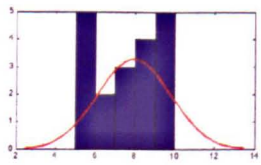
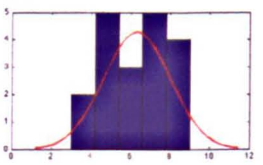
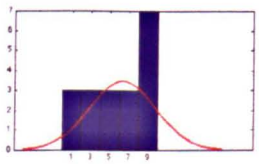
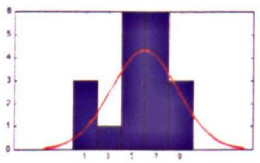
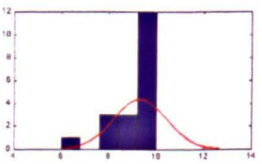
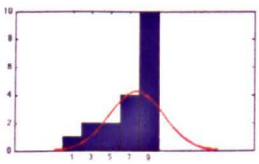
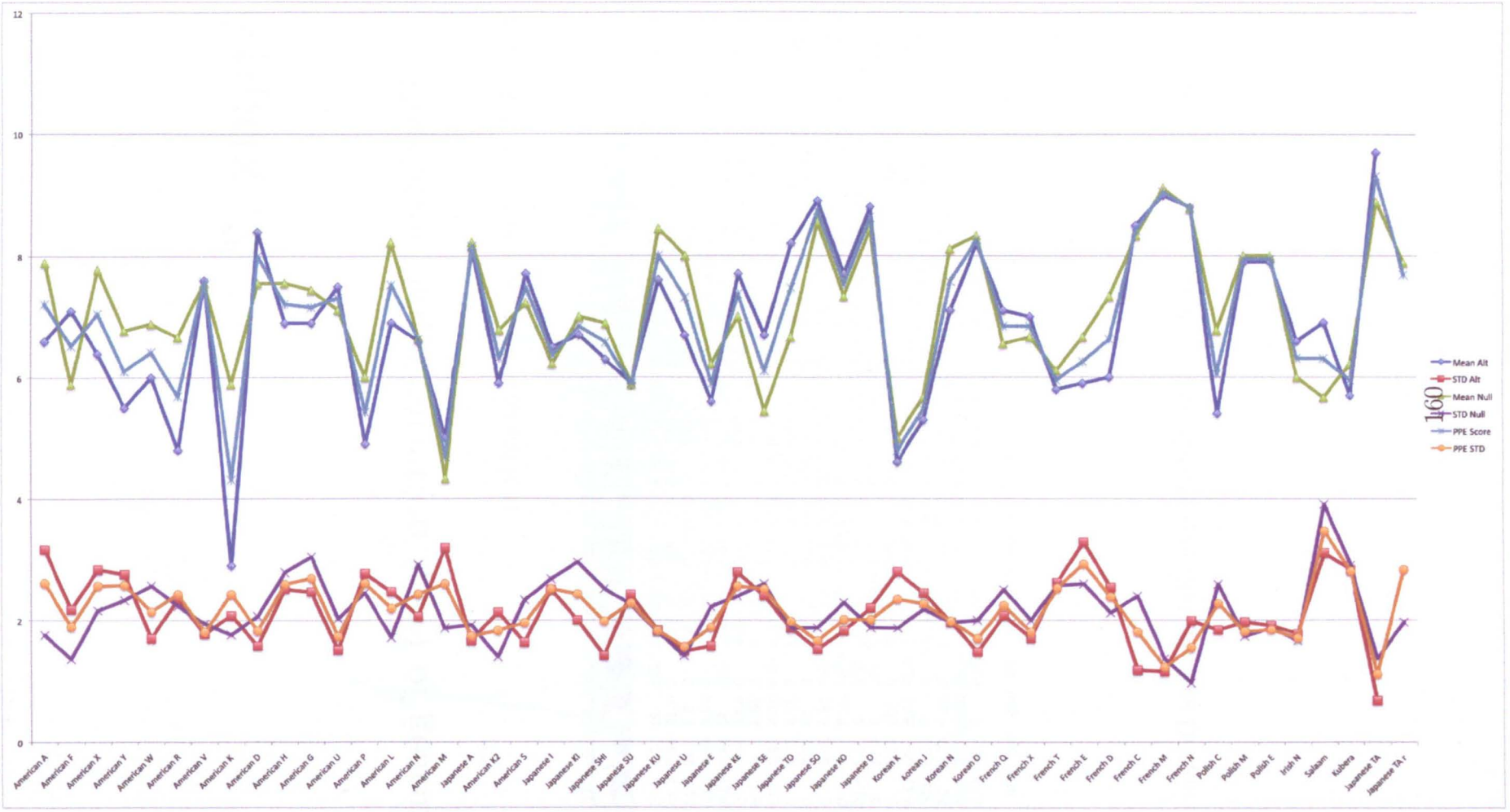
| | | |
|---|---|---|
| American S | Japanese I | Japanese KI |
|  |  |  |
| Japanese SHI | Japanese SU | Japanese KU |
|  |  |  |
| Japanese U | Japanese E | Japanese KE |
|  |  |  |
| Japanese SE | Japanese TO | Japanese SO |
|  |  |  |
| Japanese KO | Japanese O | Korean K |
|  |  |  |
| Korean J | Korean N | Korean O |
|  |  |  |

Table 8: Gaussian distribution of PPE score

| | | |
|---|---|---|
| French Q  | French X  | French T  |
| French E  | French D  | French C  |
| French M  | French N  | Polish C  |
| Polish M  | Polish E  | Irish N  |
| Salaam  | Kubera  | Japanese TA  |
| Japanese rev  | | |


12
10
8
6
4
2
0
Mean Alt
STD Alt
Mean Null
STD Null
PPE Score
PPE STD
American A
American F
American X
American Y
American W
American R
American V
American K
American D
American H
American G
American U
American P
American L
American N
American M
Japanese A
American K2
American S
Japanese I
Japanese KI
Japanese SH
Japanese SU
Japanese KU
Japanese U
Japanese E
Japanese KE
Japanese SE
Japanese TO
Japanese SO
Japanese KO
Japanese O
Korean K
Korean J
Korean N
Korean O
French Q
French X
French T
French E
French D
French C
French M
French N
Polish C
Polish M
Polish E
Irish N
Sakkam
Kushera
Japanese TA
Japanese TA r

# Appdx .2

## .2 Potential shape variation (psv) of gestures

### .2.1 Calculating shape variation

| | P_O/ | P_M/ | J_U/ | J_U/ | J_TO/ | S_B/ | A_C/ | A_O/ | A_K/ | A_H/ | point | K_N/ | A_S/ | A_A/ | J_KO/ | J_KU/ | P_E/ | thum | J_A/ | A_D/ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P_O | 100 | 17 | 4.16 | 4.16 | 41.6 | 0 | 12.5 | 12.5 | 54.2 | 25 | 37.5 | er4.1 | 16.6 | 8.3 | 50 | 12.5 | 0 | 41.6 | 37.5 | 0 |
| P_M | 16.6 | 100 | 4.16 | 37.5 | 37.5 | 66.6 | 0 | 62.5 | 16.6 | 4.16 | 20.8 | 0 | 4.16 | 0 | 4.16 | 58.3 | 4.16 | 50 | 45.8 | 16.6 |
| J_U | 4.16 | 41.6 | 100 | 91.6 | 58.3 | 50 | 0 | 54.2 | 0 | 20.8 | 0 | 0 | 8.3 | 0 | 8.3 | 50 | 25 | 50 | 0 | 41.6 |
| J_U | 4.16 | 37.5 | 91.6 | 100 | 54.2 | 62.5 | 0 | 66.6 | 8.3 | 8.3 | 8.3 | 0 | 8.3 | 0 | 20.8 | 66.6 | 29.2 | 58.3 | 8.3 | 41.6 |
| J_TO | 41.6 | 37.5 | 58.3 | 54.2 | 100 | 50 | 12.5 | 54.2 | 54.2 | 58.3 | 12.5 | 0 | 29.2 | 25 | 50 | 54.2 | 45.8 | 58.3 | 37.5 | 50 |
| S_B | 0 | 66.6 | 50 | 62.5 | 50 | 100 | 0 | 66.6 | 0 | 0 | 0 | 0 | 4.16 | 0 | 4.16 | 66.6 | 0 | 50 | 0 | 25 |
| A_C | 12.5 | 0 | 0 | 0 | 12.5 | 0 | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20.8 | 0 | 0 | 0 | 0 | 0 |
| A_O | 12.5 | 62.5 | 54.2 | 66.6 | 54.2 | 66.6 | 0 | 100 | 16.6 | 20.8 | 12.5 | 0 | 16.6 | 12.5 | 20.8 | 79.2 | 25 | 54.2 | 54.2 | 45.8 |
| A_K | 54.2 | 16.6 | 0 | 8.3 | 54.2 | 0 | 0 | 16.6 | 100 | 16.6 | 50 | 0 | 0 | 0 | 37.5 | 20.8 | 0 | 66.6 | 45.8 | 0 |
| A_H | 25 | 4.16 | 20.8 | 8.3 | 58.3 | 0 | 0 | 20.8 | 16.6 | 100 | 16.6 | 0 | 0 | 0 | 50 | 12.5 | 0 | 58.3 | 20.8 | 0 |
| point | 37.5 | 20.8 | 0 | 8.3 | 12.5 | 0 | 0 | 12.5 | 50 | 16.6 | 100 | er4.1 | 16.6 | 8.3 | 16.6 | 20.8 | 0 | 37.5 | 45.8 | 0 |
| K_N | er4.1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | er4.1 | 100 | 0 | er4.1 | 0 | 0 | 0 | 0 | 0 | 0 |
| A_S | 16.6 | 4.16 | 8.3 | 8.3 | 29.2 | 4.16 | 0 | 16.6 | 0 | 0 | 16.6 | 0 | 100 | 79.2 | 0 | 8.3 | 4.16 | 8.3 | 8.3 | 29.2 |
| A_A | 8.3 | 0 | 0 | 0 | 25 | 0 | 0 | 12.5 | 0 | 0 | 8.3 | er4.1 | 79.2 | 100 | 0 | 4.16 | 0 | 0 | 4.16 | 20.8 |
| J_KO | 50 | 4.16 | 8.3 | 20.8 | 50 | 4.2 | 20.8 | 20.8 | 37.5 | 50 | 16.6 | 0 | 0 | 0 | 100 | 20.8 | 0 | 41.6 | 20.8 | 0 |
| J_KU | 12.5 | 58.3 | 50 | 66.6 | 54.2 | 66.6 | 0 | 79.2 | 20.8 | 12.5 | 20.8 | 0 | 8.3 | 4.16 | 20.8 | 100 | 12.5 | 45.8 | 37.5 | 25 |
| P_E | 0 | 4.16 | 25 | 29.2 | 45.8 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 4.16 | 0 | 0 | 12.5 | 100 | 25 | 0 | 41.6 |
| thumb | 41.6 | 50 | 50 | 58.3 | 58.3 | 50 | 0 | 54.2 | 66.6 | 58.3 | 37.5 | 0 | 8.3 | 0 | 41.6 | 45.8 | 25 | 100 | 33.3 | 29.2 |
| J_A | 37.5 | 45.8 | 0 | 8.3 | 37.5 | 0 | 0 | 54.2 | 45.8 | 20.8 | 45.8 | 0 | 8.3 | 4.16 | 20.8 | 37.5 | 0 | 33.3 | 100 | 0 |
| A_D | 0 | 16.6 | 41.6 | 41.6 | 50 | 25 | 0 | 45.8 | 0 | 0 | 0 | 0 | 29.2 | 20.8 | 0 | 25 | 41.6 | 29.2 | 0 | 100 |
| Cluster size | vlar | mid | mid | mid | vlar | cmp | larg | larg | cmp | mid | larg | cmp | mid | cmp | larg | mid | mid | larg | cmp | mid |

| | |
|---|---|
| very large | vlar |
| large | larg |
| medium | mid |
| compact | cmp |

Figure 3: Shape resemblance calculated using Mahalanobis Distancinge

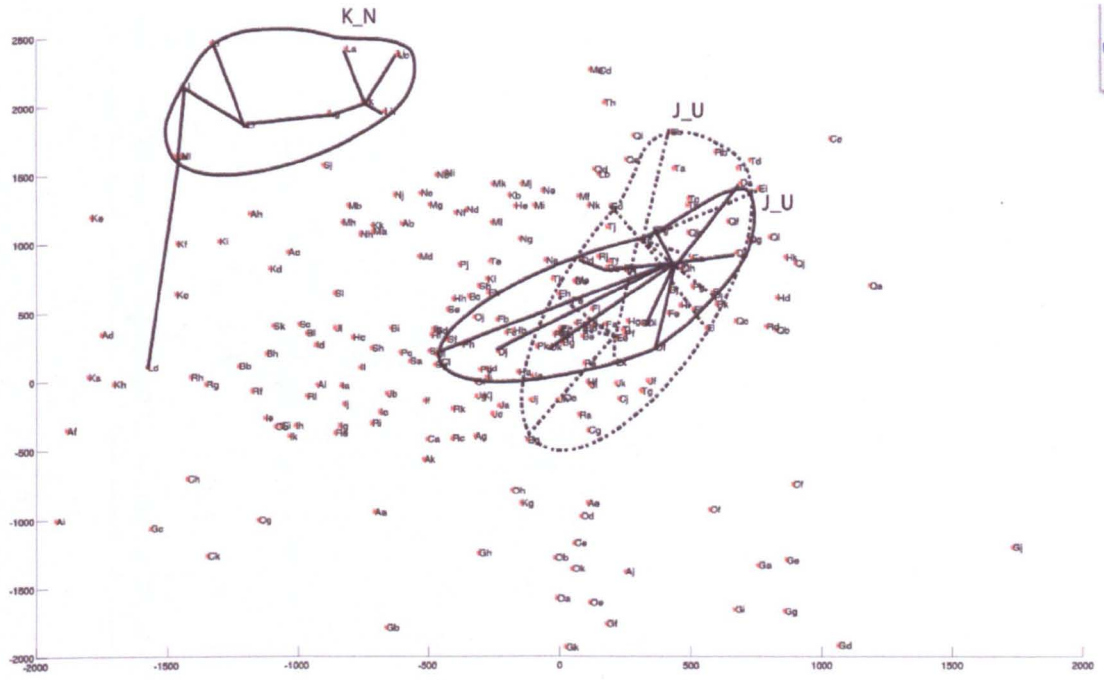



Figure 4: PCA scatterplot with Japanese U and American U labelled as J_U and Korean N labelled K_N

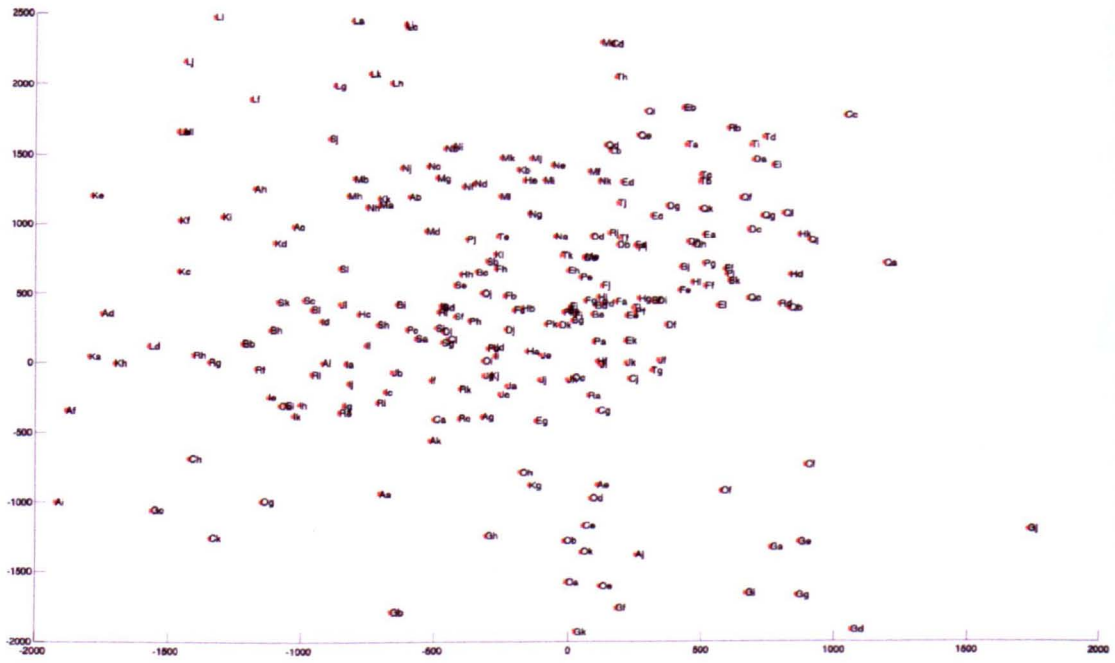



Figure 5: PCA scatterplot of all GEf postures

Table 9: PCA scatterplot (Figure 5) reference table

| Symbol | Hand dact | GiMI name | Classification number |
|---|---|---|---|
| A | Korean O | aok | 36 |
| B | Polish M | bendh | 46 |
| C | Japanes TO | clos2 | 29 |
| D | American U | dib2 | 12 |
| E | Japan U | dib | 25 |
| F | Spanish B | five | 43 |
| G | American C | nwcc | 42 |
| H | Japan O | ohh | 32 |
| I | American V | peace | 7 |
| J | French N | pistol | 44 |
| K | Japan SO | point | 30 |
| L | Korean N | pointleft | 35 |
| M | American S | power | 19 |
| N | American A | punch | 1 |
| O | Japan KO | R | 31 |
| P | Japan KU | snake | 24 |
| Q | Polish E | swan | 47 |
| R | Japanese TA | thumbs | 51 |
| S | Japanese A | thumside | 17 |
| T | American D | upone | 9 |

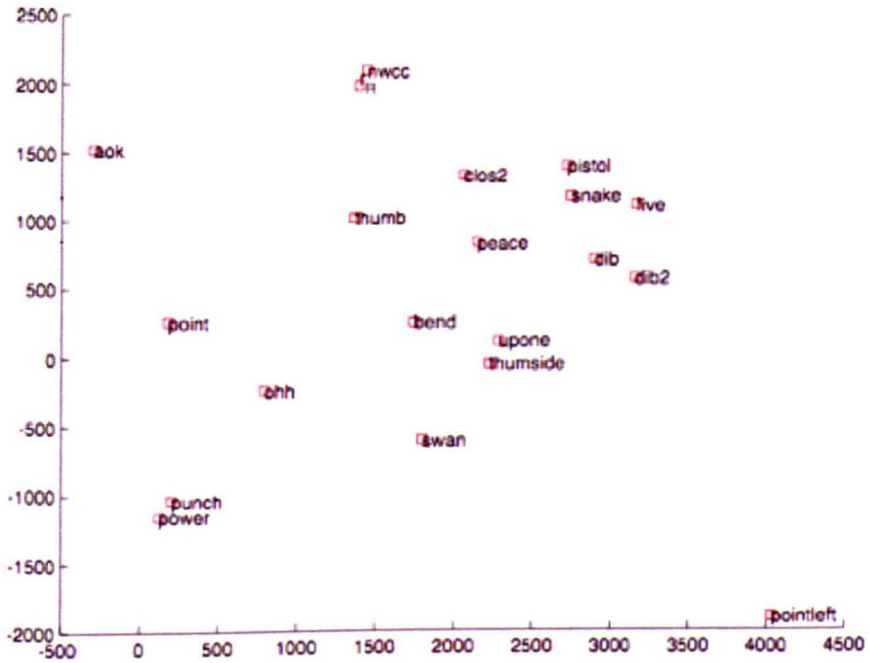



Figure 6: Shows the similarities between each GEf hand posture

# Appdx .3

## .3 Computation analysis code and components

### .3.1 Matlab analysing variance in perceived exertion score

The following code is used to calculate the mean, standard deviation and Student T test score of user study participants.

#### .3.1.1 code

```
%%%%%%%%%%
%%Load data from XML

usert = xmltree('PSV_PPE_UserData.xml');
s = convert(usert);
%s.Data;
%Patrick = [s.Data.User1.HandPostures.ASL_A; s.Data.User1.HandPostures.ASL_F];
%Patrick = [str2num(s.Data.User1.HandPostures.ASL_A) ; str2num(s.Data.User1.HandPostu
Patrick = [str2num(s.Data.User1.HandPostures.ASL_A) str2num(s.Data.User1.HandPostures

Jamie = [str2num(s.Data.User2.HandPostures.ASL_A) str2num(s.Data.User2.HandPostures.A

JaKe = [str2num(s.Data.User3.HandPostures.ASL_A) str2num(s.Data.User3.HandPostures.A
```

Sinan = [str2num(s.Data.User4.HandPostures.ASL_A) str2num(s.Data.User4.HandPostures.A

Matthew = [str2num(s.Data.User5.HandPostures.ASL_A) str2num(s.Data.User5.HandPostur

David = [str2num(s.Data.User6.HandPostures.ASL_A) str2num(s.Data.User6.HandPostures.A

Fraser = [str2num(s.Data.User7.HandPostures.ASL_A) str2num(s.Data.User7.HandPostures.A

Thomas = [str2num(s.Data.User8.HandPostures.ASL_A) str2num(s.Data.User8.HandPosture

Scheherezade = [str2num(s.Data.User9.HandPostures.ASL_A) str2num(s.Data.User9.HandPo

Emma = [str2num(s.Data.User10.HandPostures.ASL_A) str2num(s.Data.User10.HandPostur

Rose = [str2num(s.Data.User11.HandPostures.ASL_A) str2num(s.Data.User11.HandPostures

Molly = [str2num(s.Data.User12.HandPostures.ASL_A) str2num(s.Data.User12.HandPosture

Mytro = [str2num(s.Data.User13.HandPostures.ASL_A) str2num(s.Data.User13.HandPostur

Lauren = [str2num(s.Data.User14.HandPostures.ASL_A) str2num(s.Data.User14.HandPostur

Abigail = [str2num(s.Data.User15.HandPostures.ASL_A) str2num(s.Data.User15.HandPostur

Sanya = [str2num(s.Data.User16.HandPostures.ASL_A) str2num(s.Data.User16.HandPostures

Charlotte = [str2num(s.Data.User17.HandPostures.ASL_A) str2num(s.Data.User17.HandPost

Sophie = [str2num(s.Data.User18.HandPostures.ASL_A) str2num(s.Data.User18.HandPostur

Kate = [str2num(s.Data.User19.HandPostures.ASL_A) str2num(s.Data.User19.HandPostures

```
%%%%%%%
%% Initialise variables

pixhigh = 255;
nullg = 9;
altg =10;

dactNum = 19;
gesturePPEScore = zeros(52, 19, 'double')
mahalVal = zeros(19, 19, 'uint32')
testNull = zeros(19, 19, 'uint32')
testNulImage = zeros(19, 19, 'uint32')
cutNullImage = zeros(nullg, nullg, 'uint32')
AltNulImage = zeros(altg, altg, 'uint32')
FinalNulImage = zeros(52, 38, 'uint32')
splitCompareImage = zeros(52,52 , 'uint32')

GestNull = zeros(52, 52, 'uint32')
GestNulImage = zeros(52, 52, 'uint32')
GestpVal = zeros(52, 52, 'double')
GestmahalVal = zeros(52, 52, 'uint32')
GestSingleTestNulImage = zeros(52, 52, 'uint32')

algGestureSTD = zeros(52, 1, 'double')
nullgGestureSTD = zeros(52, 1, 'double')

testpVal = zeros(19, 19, 'double')
testConfI= zeros(19, 19, 'uint32')
femalePPEScore = zeros(52, 11, 'uint32')
trimfemalePPEScore = zeros(52, 8, 'uint32')
malePPEScore = zeros(52, 8, 'uint32')
gestureMeanScore = zeros(52, 1, 'double')
gestureSTDScore = zeros(52, 1, 'double')
```

```
fgestureMeanScore = zeros(52, 1, 'double')
mgestureSTDScore = zeros(52, 1, 'double')
mgestureMeanScore = zeros(52, 1, 'double')
fgestureSTDScore = zeros(52, 1, 'double')
%%%%%%%% Create table to display data


altgGestureScore = zeros( 52,altg, 'double')
nullgGestureScore = zeros( 52, nullg,'double')

%%%%%
%% Assign memory for arrays / matrices

for i = 1:52,
gesturePPEScore(i,1:end)=[Abigail(i) Charlotte(i) David(i) Emma(i) Fraser(i) JaKe(i) Jamie
Rose(i) Sanya(i) Scheherezade(i) Sinan(i) Sophie(i) Thomas(i)]
femalePPEScore(i,1:end)=[Abigail(i) Charlotte(i) Emma(i) Kate(i) Lauren(i) Molly(i) Mytro
malePPEScore(i,1:end)=[David(i) Fraser(i) JaKe(i) Jamie(i) Matthew(i) Patrick(i) Sinan(i)
trimfemalePPEScore(i,1:end)=[Abigail(i) Charlotte(i) Emma(i) Kate(i) Lauren(i) Molly(i) M
fgestureSTDScore(i,1:end) = std2(femalePPEScore(i,1:end))
mgestureSTDScore(i,1:end) = std2(malePPEScore(i,1:end))
fgestureMeanScore(i,1:end) = mean(femalePPEScore(i,1:end))
mgestureMeanScore(i,1:end) = mean(malePPEScore(i,1:end))
gestureSTDScore(i,1:end) = std2(gesturePPEScore(i,1:end))
gestureMeanScore(i,1:end) = mean(gesturePPEScore(i,1:end))
% b = strread(num2str(gesturePPEScore(i,1:end)),'%f')
end


User = cat( 19, [Abigail], [Charlotte], [Emma], [Kate], [Lauren], [Molly], [Mytro], [Rose], [San
UserNull = cat( nullg, [Emma], [Kate], [Molly], [Rose], [Sanya], [Sophie], [David], [Matthew],
UserAlt = cat( altg, [Abigail], [Charlotte], [Lauren], [Mytro], [Scheherezade], [Fraser], [JaKe],
[Thomas])
```

```
%%%%% (Window) inter cutnull group ttest grid
%% Compare each member of the (h0) user set and map into image


for i = 1:nullg,

 Y= UserNull(:,:,i);

 for k =1:nullg,
 X= UserNull(:,:,k);
 [h,p] = ttest2(Y, X);
 end

 if h == 1

    cutNulImage(i,k) = pixhigh;
   else
    cutNulImage(i,k) = 0;
 end
end
%%%%%%%%%%%%%%%%%
%% Definitive Null calculation the mean of each posture calculated across
%% the whole user sample
%for k = 1:19,
for i = 1:19,
 for j = 1:52,
%Y= gesturePPEScore(j,i);
X= gesturePPEScore(j,k);
%[h,p] = ttest2(Y, X);
[h,p] = ttest(X);
 if h == 1
    FinalNulImage(j,i) = pixhigh;
```

```
   else
    FinalNulImage(j,i) = 0;
 end

 end
end
%end


%%%%%%%%%% (window) inter Altnull group ttest grid
%% Compare each member of the (h1) user set and map into image


for i = 1:altg,

 Y= UserAlt(:,:,i);

 for k =1:altg,
 X= UserAlt(:,:,k);
 [h,p] = ttest2(Y, X);
 end
 if h == 1
    AltNulImage(i,k) = pixhigh;
   else
    AltNulImage(i,k) = 0;
 end
end

%%%%%%%%
%% Find the STD of halved test sample

for i = 1:altg,
 for k =1:52,
```

```
 altgGestureScore(k,i) = UserAlt(:,k,i);
 end

end

for i = 1:nullg,
 for k =1:52,
  nullgGestureScore(k,i) = UserNull(:,k,i);

 end

end
for k =1:52,
algGestureSTD(k,1:end) = std2(altgGestureScore(k,1:end));
nullgGestureSTD(k,1:end) = std2(nullgGestureScore(k,1:end));
end
%%%%%%%% (Window) Compare split set
%% Create image map that compare the interrelation between Seperated sample

for i = 1:52,
 for k = 1:52,
  h = ttest2(nullgGestureScore(i,:,1,end),altgGestureScore(k,:,1,end));
  if h == 1
    splitCompareImage(i,k) = pixhigh;
   else
    splitCompareImage(i,k) = 0;
  end
 end
end

%%%%%%%%%
%% Calculate the mahalanobis distance across sample and do a ttest across
%% whole sample
```

```
for i = 1:19,

  Y = User(:,:,i);

 for k = 1:19,

  X = User(:,:,k);
  nx = size(X, 1); ny = size(Y, 1); m = mean(X); C = cov(X); %mahalanobis calculation
  d = zeros(ny, 1); %mahalanobis calculation

   for j = 1:ny %mahalanobis calculation
   %size of set in X % size of set in Y
   d(j) = (Y(j,:) - m) / C * (Y(j,:) - m)';
   end
   [h,p] = ttest2(Y, X);
   testNull(i,k) = h; %t-test null hypothesis that there is no significant discrepencies betwe


%%%%%%%%%%% (Window ) inter group ttest grid
   if h == 1
    pixhigh = 255;
    testNullImage(i,k) = pixhigh;
   else
    testNullImage(i,k) = 0;
   end

   testpVal(i,k) = p; %t-test p value
   % testConfI(i,k)= ci; % confidence interval
   mahalVal(i,k) = d; %Calculate the mahalanobis distance between participants PPEscore
```

```
 end

end

%Gest = gesturePPEScore(i,1:end);
%[hges,pges]=ttest2(Gest);


%%%%%%%%%
%% Create a gesture PPE score matrix


AmericanA = cat( 19, [Abigail(1)], [Charlotte(1)], [Emma(1)], [Kate(1)], [Lauren(1)], [Molly(
AmericanF = cat( 19, [Abigail(2)], [Charlotte(2)], [Emma(2)], [Kate(2)], [Lauren(2)], [Molly(
AmericanX = cat( 19, [Abigail(3)], [Charlotte(3)], [Emma(3)], [Kate(3)], [Lauren(3)], [Molly(
AmericanY = cat( 19, [Abigail(4)], [Charlotte(4)], [Emma(4)], [Kate(4)], [Lauren(4)], [Molly(
AmericanW = cat( 19, [Abigail(5)], [Charlotte(5)], [Emma(5)], [Kate(5)], [Lauren(5)], [Molly(
AmericanR = cat( 19, [Abigail(6)], [Charlotte(6)], [Emma(6)], [Kate(6)], [Lauren(6)], [Molly(
AmericanV = cat( 19, [Abigail(7)], [Charlotte(7)], [Emma(7)], [Kate(7)], [Lauren(7)], [Molly(
AmericanK = cat( 19, [Abigail(8)], [Charlotte(8)], [Emma(8)], [Kate(8)], [Lauren(8)], [Molly(
AmericanD = cat( 19, [Abigail(9)], [Charlotte(9)], [Emma(9)], [Kate(9)], [Lauren(9)], [Molly(
AmericanH = cat( 19, [Abigail(10)], [Charlotte(10)], [Emma(10)], [Kate(10)], [Lauren(10)], [M
AmericanG = cat( 19, [Abigail(11)], [Charlotte(11)], [Emma(11)], [Kate(11)], [Lauren(11)], [M
AmericanU = cat( 19, [Abigail(12)], [Charlotte(12)], [Emma(12)], [Kate(12)], [Lauren(12)], [M
AmericanP = cat( 19, [Abigail(13)], [Charlotte(13)], [Emma(13)], [Kate(13)], [Lauren(13)], [M
AmericanL = cat( 19, [Abigail(14)], [Charlotte(14)], [Emma(14)], [Kate(14)], [Lauren(14)], [M
AmericanN = cat( 19, [Abigail(15)], [Charlotte(15)], [Emma(15)], [Kate(15)], [Lauren(15)], [M
AmericanM = cat( 19, [Abigail(16)], [Charlotte(16)], [Emma(16)], [Kate(16)], [Lauren(16)], [M
JapaneseA = cat( 19, [Abigail(17)], [Charlotte(17)], [Emma(17)], [Kate(17)], [Lauren(17)], [M
AmericanK2 = cat( 19, [Abigail(18)], [Charlotte(18)], [Emma(18)], [Kate(18)], [Lauren(18)], [
AmericanS = cat( 19, [Abigail(19)], [Charlotte(19)], [Emma(19)], [Kate(19)], [Lauren(19)], [M
JapaneseI = cat( 19, [Abigail(20)], [Charlotte(20)], [Emma(20)], [Kate(20)], [Lauren(20)], [M
JapaneseKI = cat( 19, [Abigail(21)], [Charlotte(21)], [Emma(21)], [Kate(21)], [Lauren(21)], [M
```

JapaneseSHI = cat( 19, [Abigail(22)], [Charlotte(22)], [Emma(22)], [Kate(22)], [Lauren(22)],
JapaneseSU = cat( 19, [Abigail(23)], [Charlotte(23)], [Emma(23)], [Kate(23)], [Lauren(23)],
JapaneseKU = cat( 19, [Abigail(24)], [Charlotte(24)], [Emma(24)], [Kate(24)], [Lauren(24)],
JapaneseU = cat( 19, [Abigail(25)], [Charlotte(25)], [Emma(25)], [Kate(25)], [Lauren(25)], [M
JapaneseE = cat( 19, [Abigail(26)], [Charlotte(26)], [Emma(26)], [Kate(26)], [Lauren(26)], [M
JapaneseKE = cat( 19, [Abigail(27)], [Charlotte(27)], [Emma(27)], [Kate(27)], [Lauren(27)],
JapaneseSE = cat( 19, [Abigail(28)], [Charlotte(28)], [Emma(28)], [Kate(28)], [Lauren(28)],
JapaneseTO = cat( 19, [Abigail(29)], [Charlotte(29)], [Emma(29)], [Kate(29)], [Lauren(29)],
JapaneseSO = cat( 19, [Abigail(30)], [Charlotte(30)], [Emma(30)], [Kate(30)], [Lauren(30)],
JapaneseKO = cat( 19, [Abigail(31)], [Charlotte(31)], [Emma(31)], [Kate(31)], [Lauren(31)],
JapaneseO = cat( 19, [Abigail(32)], [Charlotte(32)], [Emma(32)], [Kate(32)], [Lauren(32)], [M
KoreanK = cat( 19, [Abigail(33)], [Charlotte(33)], [Emma(33)], [Kate(33)], [Lauren(33)], [Mo
KoreanJ = cat( 19, [Abigail(34)], [Charlotte(34)], [Emma(34)], [Kate(34)], [Lauren(34)], [Mol
KoreanN = cat( 19, [Abigail(35)], [Charlotte(35)], [Emma(35)], [Kate(35)], [Lauren(35)], [Mo
KoreanO = cat( 19, [Abigail(36)], [Charlotte(36)], [Emma(36)], [Kate(36)], [Lauren(36)], [Mo
FrenchQ = cat( 19, [Abigail(37)], [Charlotte(37)], [Emma(37)], [Kate(37)], [Lauren(37)], [Mol
FrenchX = cat( 19, [Abigail(38)], [Charlotte(38)], [Emma(38)], [Kate(38)], [Lauren(38)], [Mol
FrenchT = cat( 19, [Abigail(39)], [Charlotte(39)], [Emma(39)], [Kate(39)], [Lauren(39)], [Mol
FrenchE = cat( 19, [Abigail(40)], [Charlotte(40)], [Emma(40)], [Kate(40)], [Lauren(40)], [Mol
FrenchD = cat( 19, [Abigail(41)], [Charlotte(41)], [Emma(41)], [Kate(41)], [Lauren(41)], [Mol
FrenchC = cat( 19, [Abigail(42)], [Charlotte(42)], [Emma(42)], [Kate(42)], [Lauren(42)], [Mol
FrenchM = cat( 19, [Abigail(43)], [Charlotte(43)], [Emma(43)], [Kate(43)], [Lauren(43)], [Mol
FrenchN = cat( 19, [Abigail(44)], [Charlotte(44)], [Emma(44)], [Kate(44)], [Lauren(44)], [Mol
PolishC = cat( 19, [Abigail(45)], [Charlotte(45)], [Emma(45)], [Kate(45)], [Lauren(45)], [Moll
PolishM = cat( 19, [Abigail(46)], [Charlotte(46)], [Emma(46)], [Kate(46)], [Lauren(46)], [Moll
PolishE = cat( 19, [Abigail(47)], [Charlotte(47)], [Emma(47)], [Kate(47)], [Lauren(47)], [Molly
IrishN = cat( 19, [Abigail(48)], [Charlotte(48)], [Emma(48)], [Kate(48)], [Lauren(48)], [Molly
Salaam = cat( 19, [Abigail(49)], [Charlotte(49)], [Emma(49)], [Kate(49)], [Lauren(49)], [Molly
Kubera = cat( 19, [Abigail(50)], [Charlotte(50)], [Emma(50)], [Kate(50)], [Lauren(50)], [Molly
JapaneseTA = cat( 19, [Abigail(51)], [Charlotte(51)], [Emma(51)], [Kate(51)], [Lauren(51)],
JapaneseTArev = cat( 19, [Abigail(52)], [Charlotte(52)], [Emma(52)], [Kate(52)], [Lauren(52)

% Gest = gesturePPEScore(i,:);

```
%[hges,pges]=ttest(Gest);


Gestures = cat( 52,[AmericanA(:)], [AmericanF(:)],[AmericanX(:)], [AmericanY(:)] ,[America

%%%%%%
%% Build image to compare the variance of PPE score

for i = 1:52,

  Y = Gestures(:,:,i);

 for k = 1:52,

  X = Gestures(:,:,k);
  nx = size(X, 1); ny = size(Y, 1); m = mean(X); C = cov(X); %mahalanobis calculation
  d = zeros(ny, 1); %mahalanobis calculation

   for j = 1:ny %mahalanobis calculation
   %size of set in X % size of set in Y
   d(j) = (Y(j,:) - m) / C * (Y(j,:) - m)';
   end
   %[h,p,ci,stats] = ttest2(Y, X,[],[],'unequal');
   [h] = ttest2(Y, X,[],[],'unequal');
   %testNull(i,k) = h; %t-test null hypothesis that there is no significant discrepencies betw
   % [h2,p2,ci2,stats2]=ttest(Y)
   [h2] = ttest(Y);

   if h == 1
    pixhigh = 255;
    GestNullImage(i,k) = pixhigh;

   else
```

```
    GestNulImage(i,k) = 0;
   end
   if h2 == 1

    GestSingleTestNulImage(i,k) = pixhigh;
   else
    GestSingleTestNulImage(i,k) = 0;
   end

   % GestpVal(i,k) = p; %t-test p value
   % testConfI(i,k)= ci; % confidence interval
   % GestmahalVal(i,k) = d; %Calculate the mahalanobis distance between participants PPE
 end



end

%%%%%%%
%% Display images

figure('name','Gesture ttest grid');
colormap(gray)
image(GestNulImage)

figure('name','Definitive ttest grid');
colormap(gray)
image(FinalNulImage)


figure('name','Compare split set');
colormap(gray)
image(splitCompareImage)
```

```
figure('name','Gesture Mahalanobis');
colormap(cool)
image(GestmahalVal)

figure('name','Gesture Single ttest grid');
colormap(gray)
image(GestSingleTestNulImage)

figure('name','inter group ttest grid');
colormap(hot)
image(testNulImage)

figure('name','inter cutnull group ttest grid');
colormap(gray)
image(cutNulImage)

figure('name','inter Altnull group ttest grid');
colormap(gray)
image(AltNulImage)

figure('name','Mahalanobis');
colormap(cool)
image(mahalVal)
%write the column headers first
S3 = cat(2, gestureMeanScore , gestureSTDScore);
S4 = cat(2, fgestureMeanScore , mgestureMeanScore);
S5 = cat(2, fgestureSTDScore , mgestureSTDScore);
S6 = cat(1, testNull);
S7 = cat(1, testpVal);
S8 = cat(1, mahalVal);
S9 = cat(2, altgGestureScore, nullgGestureScore);
```

```
S10= cat(2, algGestureSTD, nullgGestureSTD);



%%%%%%%%%
%write the data directly underneath the column headers
%% Create cSV / XSL spreadsheets
xlswrite('GEfPPEScoreAv.xls', S3 );
xlswrite('GEfGenderScoreDiff.xls', S4 );
xlswrite('GEfGenderSTDDiff.xls', S5 );
xlswrite('GEfUserttest.xls', S6 );
xlswrite('GEfUserpval.xls', S7 );
xlswrite('GEfMahal.xls', S8 );
xlswrite('GroupSplit.xls', S9 );
xlswrite('GroupSplitSTD.xls', S10 );



%h = ttest2(trimfemalePPEScore, malePPEScore);
%xlswrite('myDataFile.xls', gestureSTDScore,'Sheet1','A2');
%xlswrite('myDataFile.xls', gestureMeanScore,'Sheet1','A3');
%%%%%%%%%%%% t(1,1:end) this command will display the array of ppe
%%%%%%%%%%%% score the American A posture
%%% b = strread(num2str(gesturePPEScore(i,1:end)),'%f') Convert to
%%% float
%% histfit(b) display normal distribution histograph
%%%%
%%
```

## .3.2 Matlab analysing Potential Shape difference using PCA

The following code was written in order to calculate the range of shape variance exhibited by a broad range of people replicating specific gestures. Since patterns

in data can be hard to find in data of high dimension PCA has been used.

### .3.2.1 code

```
function a2_current_update_makeAll_scatterplot(filelist)

%%Plots comparative PCA values of the GEF dataset
%
%
%To Execute type a1_current_update_pca_hands into terminal
%


% Images = [];
w=40; h=40;

% Open input file
fprintf(1,'Read directory name.\n');

numimgs = 11;
% fid = fopen('filelist','r');
aok_id = fopen('filelists/aok_filelist','r');
bend_id = fopen('filelists/bendh_filelist','r');
clos_id = fopen('filelists/clos2_filelist','r');
dib_id = fopen('filelists/dib_filelist','r');
dib2n_id = fopen('filelists/dib2nd_filelist','r');
five_id = fopen('filelists/five_filelist','r');
nwcc_id = fopen('filelists/nwcc_filelist','r');
ohh_id = fopen('filelists/ohh_filelist','r');
peace_id = fopen('filelists/peace_filelist','r');
pistol_id = fopen('filelists/pistol_filelist','r');
point_id = fopen('filelists/point_R_filelist','r');
```

```
pointL_id = fopen('filelists/pointleft_filelist','r');
power_id = fopen('filelists/power_filelist','r');
punch_id = fopen('filelists/punch_filelist','r');
r_id = fopen('filelists/r_filelist','r');
snake_id = fopen('filelists/snake_filelist','r');
swan_id = fopen('filelists/swan_filelist','r');
thumb_id = fopen('filelists/thumbs_filelist','r');
thumbsid_id = fopen('filelists/thumbside_filelist','r');
upone_id = fopen('filelists/upone_filelist','r');


%%//////////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////////
if aok_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
aok_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(aok_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 aok_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(aok_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
```

```
%%//////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////


 if bend_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
 end;

 % Get the images
bend_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(bend_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 bend_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(bend_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////


 if clos_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
clos_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
```

```
for i = 1:numimgs
 imgname = fgetl(clos_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 clos_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(clos_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////////
if dib_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
dib_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(dib_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 dib_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(dib_id); % Close the filelist when done
```

```
fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////
if dib2n_id < 0 || numimgs < 1
error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
dib2n_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(dib2n_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 dib2n_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(dib2n_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////
if five_id < 0 || numimgs < 1
error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
five_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
```

```
for i = 1:numimgs
 imgname = fgetl(five_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 five_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(five_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////////
 if nwcc_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
nwcc_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(nwcc_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 nwcc_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(nwcc_id); % Close the filelist when done
```

```
fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////
 if ohh_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
ohh_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(ohh_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 ohh_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(ohh_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////
 if peace_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
peace_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
```

```
for i = 1:numimgs
 imgname = fgetl(peace_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 peace_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(peace_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////////
 if pistol_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
pistol_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(pistol_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 pistol_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(pistol_id); % Close the filelist when done
```

```
fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////////
if point_id < 0 || numimgs < 1
error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
point_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
imgname = fgetl(point_id);
fprintf(1,'loading PGM file %s\n',imgname);
%writepgm(imgname); % Read this image as a 2D array
%
Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
point_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(point_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////////
if pointL_id < 0 || numimgs < 1
error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
pointL_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
```

```
for i = 1:numimgs
 imgname = fgetl(pointL_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 pointL_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(pointL_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////////
 if power_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
power_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(power_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 power_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(power_id); % Close the filelist when done
```

```
fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////
 if punch_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
punch_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(punch_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 punch_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(punch_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////
 if r_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
r_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
```

```
for i = 1:numimgs
 imgname = fgetl(r_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 r_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(r_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////////
 if snake_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
snake_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(snake_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 snake_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(snake_id); % Close the filelist when done
```

```
fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////////

if swan_id < 0 || numimgs < 1
error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
swan_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(swan_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 swan_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(swan_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////////

if thumb_id < 0 || numimgs < 1
error(['Cannot get list of images from file "' filelist, '"']);
end;
```

```
% Get the images
thumb_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(thumb_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 thumb_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(thumb_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%//////////////////////////////////////////////////////////////////////////
%%//////////////////////////////////////////////////////////////////////////

 if thumbsid_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
thumbsid_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(thumbsid_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
% Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 thumbsid_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
```

```
end;
fclose(thumbsid_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////
 if upone_id < 0 || numimgs < 1
 error(['Cannot get list of images from file "' filelist, '"']);
end;

% Get the images
upone_Images = zeros(w*h,numimgs); % - preallocate size of the return matrix
for i = 1:numimgs
 imgname = fgetl(upone_id);
 fprintf(1,'loading PGM file %s\n',imgname);
 %writepgm(imgname); % Read this image as a 2D array
 %
 Img = imread(imgname);
 % Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
 upone_Images(1:w*h,i) = reshape(Img',w*h,1); % Make a column vector
end;
fclose(upone_id); % Close the filelist when done

fprintf(1,'Read %d images.\n',numimgs);

% The function returns the output arguments Images, w, and h here.
%%////////////////////////////////////////////////////////////////////
%%////////////////////////////////////////////////////////////////////
[aok_Vecs,aok_Vals,aok_Psi] = pc_evectors(aok_Images,11)
[bend_Vecs,bend_Vals,aok_Psi] = pc_evectors(bend_Images,11)
[clos_Vecs,clos_Vals,clos_Psi] = pc_evectors(clos_Images,11)
```

```
[dib_Vecs,dib_Vals,dib_Psi] = pc_evectors(dib_Images,11)
[dib2n_Vecs,dib2n_Vals,dib2n_Psi] = pc_evectors(dib2n_Images,11)
[five_Vecs,five_Vals,five_Psi] = pc_evectors(five_Images,11)
[nwcc_Vecs,nwcc_Vals,nwcc_Psi] = pc_evectors(nwcc_Images,11)
[ohh_Vecs,ohh_Vals,ohh_Psi] = pc_evectors(ohh_Images,11)
[peace_Vecs,peace_Vals,peace_Psi] = pc_evectors(peace_Images,11)
[pistol_Vecs,pistol_Vals,pistol_Psi] = pc_evectors(pistol_Images,11)
[point_Vecs,point_Vals,point_Psi] = pc_evectors(point_Images,11)
[pointL_Vecs,pointL_Vals,pointL_Psi] = pc_evectors(pointL_Images,11)
[power_Vecs,power_Vals,power_Psi] = pc_evectors(power_Images,11)
[punch_Vecs,punch_Vals,punch_Psi] = pc_evectors(punch_Images,11)
[r_Vecs,r_Vals,r_Psi] = pc_evectors(r_Images,11)
[snake_Vecs,snake_Vals,snake_Psi] = pc_evectors(snake_Images,11)
[swan_Vecs,swan_Vals,swan_Psi] = pc_evectors(swan_Images,11)
[thumb_Vecs,thumb_Vals,thumb_Psi] = pc_evectors(thumb_Images,11)
[thumbsid_Vecs,thumbsid_Vals,thumbsid_Psi] = pc_evectors(thumbsid_Images,11)
[upone_Vecs,upone_Vals,upone_Psi] = pc_evectors(upone_Images,11)

%%////////////////////////////////////////////////////////////////////////

%plot(aok_Vals);
aok_CVals = zeros(1,length(aok_Vals));
aok_CVals(1) = aok_Vals(1);
for i = 2:length(aok_Vals) % Accumulate the eigenvalue sum
aok_CVals(i) = aok_CVals(i-1) + aok_Vals(i);
end;
aok_CVals = aok_CVals / sum(aok_Vals);
%%////////////////////////////////////////////////////////////////////////
%plot(bend_Vals);
bend_CVals = zeros(1,length(bend_Vals));
bend_CVals(1) = bend_Vals(1);
for i = 2:length(bend_Vals) % Accumulate the eigenvalue sum
bend_CVals(i) = bend_CVals(i-1) + bend_Vals(i);
```

```
end;
bend_CVals = bend_CVals / sum(bend_Vals);
%%//////////////////////////////////////////////////////////////////////

%plot(clos_Vals);
clos_CVals = zeros(1,length(clos_Vals));
clos_CVals(1) = clos_Vals(1);
for i = 2:length(clos_Vals) % Accumulate the eigenvalue sum
clos_CVals(i) = clos_CVals(i-1) + clos_Vals(i);
end;
clos_CVals = clos_CVals / sum(clos_Vals);
%%//////////////////////////////////////////////////////////////////////
%plot(dib_Vals);
dib_CVals = zeros(1,length(dib_Vals));
dib_CVals(1) = dib_Vals(1);
for i = 2:length(dib_Vals) % Accumulate the eigenvalue sum
dib_CVals(i) = dib_CVals(i-1) + dib_Vals(i);
end;
dib_CVals = dib_CVals / sum(dib_Vals);
%%//////////////////////////////////////////////////////////////////////

%plot( dib2n_Vals);
dib2n_CVals = zeros(1,length( dib2n_Vals));
dib2n_CVals(1) = dib2n_Vals(1);
for i = 2:length( dib2n_Vals) % Accumulate the eigenvalue sum
dib2n_CVals(i) = dib2n_CVals(i-1) + dib2n_Vals(i);
end;
dib2n_CVals = dib2n_CVals / sum( dib2n_Vals);
%%//////////////////////////////////////////////////////////////////////
%plot(five_Vals);
five_CVals = zeros(1,length(five_Vals));
five_CVals(1) = five_Vals(1);
for i = 2:length(five_Vals) % Accumulate the eigenvalue sum
```

```
five_CVals(i) = five_CVals(i-1) + five_Vals(i);
end;
five_CVals = five_CVals / sum(five_Vals);
%%//////////////////////////////////////////////////////////////////////////
%plot(nwcc_Vals);
nwcc_CVals = zeros(1,length(nwcc_Vals));
nwcc_CVals(1) = nwcc_Vals(1);
for i = 2:length(nwcc_Vals) % Accumulate the eigenvalue sum
nwcc_CVals(i) = nwcc_CVals(i-1) + nwcc_Vals(i);
end;
nwcc_CVals = nwcc_CVals / sum(nwcc_Vals);
%%//////////////////////////////////////////////////////////////////////////
% plot(ohh_Vals);
ohh_CVals = zeros(1,length(ohh_Vals));
ohh_CVals(1) = ohh_Vals(1);
for i = 2:length(ohh_Vals) % Accumulate the eigenvalue sum
ohh_CVals(i) = ohh_CVals(i-1) + ohh_Vals(i);
end;
ohh_CVals = ohh_CVals / sum(ohh_Vals);
%%//////////////////////////////////////////////////////////////////////////
% plot(peace_Vals);
peace_CVals = zeros(1,length(peace_Vals));
peace_CVals(1) = peace_Vals(1);
for i = 2:length(peace_Vals) % Accumulate the eigenvalue sum
peace_CVals(i) = peace_CVals(i-1) + peace_Vals(i);
end;
peace_CVals = peace_CVals / sum(peace_Vals);
%%//////////////////////////////////////////////////////////////////////////
%plot(pistol_Vals);
pistol_CVals = zeros(1,length(pistol_Vals));
pistol_CVals(1) = pistol_Vals(1);
for i = 2:length(pistol_Vals) % Accumulate the eigenvalue sum
pistol_CVals(i) = pistol_CVals(i-1) + pistol_Vals(i);
```

```
end;
pistol_CVals = pistol_CVals / sum(pistol_Vals);
%%//////////////////////////////////////////////////////////////////////
 %plot(pointL_Vals);
pointL_CVals = zeros(1,length(pointL_Vals));
pointL_CVals(1) = pointL_Vals(1);
for i = 2:length(pointL_Vals) % Accumulate the eigenvalue sum
pointL_CVals(i) = pointL_CVals(i-1) + pointL_Vals(i);
end;
pointL_CVals = pointL_CVals / sum(pointL_Vals);
%%//////////////////////////////////////////////////////////////////////
 %plot(power_Vals);
power_CVals = zeros(1,length(power_Vals));
power_CVals(1) = power_Vals(1);
for i = 2:length(power_Vals) % Accumulate the eigenvalue sum
power_CVals(i) = power_CVals(i-1) + power_Vals(i);
end;
power_CVals = power_CVals / sum(power_Vals);
%%//////////////////////////////////////////////////////////////////////
 %plot(punch_Vals);
punch_CVals = zeros(1,length(punch_Vals));
punch_CVals(1) = punch_Vals(1);
for i = 2:length(punch_Vals) % Accumulate the eigenvalue sum
punch_CVals(i) = punch_CVals(i-1) + punch_Vals(i);
end;
punch_CVals = punch_CVals / sum(punch_Vals);
%%//////////////////////////////////////////////////////////////////////
 %plot(r_Vals);
r_CVals = zeros(1,length(r_Vals));
r_CVals(1) = r_Vals(1);
for i = 2:length(r_Vals) % Accumulate the eigenvalue sum
r_CVals(i) = r_CVals(i-1) + r_Vals(i);
end;
```

```
r_CVals = r_CVals / sum(r_Vals);
%%//////////////////////////////////////////////////////////////////////////
 % plot(r_Vals);
snake_CVals = zeros(1,length(snake_Vals));
snake_CVals(1) = snake_Vals(1);
for i = 2:length(snake_Vals) % Accumulate the eigenvalue sum
snake_CVals(i) = snake_CVals(i-1) + snake_Vals(i);
end;
snake_CVals = snake_CVals / sum(snake_Vals);
%%//////////////////////////////////////////////////////////////////////////
 % plot(swan_Vals);
swan_CVals = zeros(1,length(swan_Vals));
swan_CVals(1) = swan_Vals(1);
for i = 2:length(swan_Vals) % Accumulate the eigenvalue sum
swan_CVals(i) = swan_CVals(i-1) + swan_Vals(i);
end;
swan_CVals = swan_CVals / sum(swan_Vals);
%%//////////////////////////////////////////////////////////////////////////
 % plot(thumb_Vals);
thumb_CVals = zeros(1,length(thumb_Vals));
thumb_CVals(1) = thumb_Vals(1);
for i = 2:length(thumb_Vals) % Accumulate the eigenvalue sum
thumb_CVals(i) = thumb_CVals(i-1) + thumb_Vals(i);
end;
thumb_CVals = thumb_CVals / sum(thumb_Vals);
%%//////////////////////////////////////////////////////////////////////////
 %plot(thumbsid_Vals);
thumbsid_CVals = zeros(1,length(thumbsid_Vals));
thumbsid_CVals(1) = thumbsid_Vals(1);
for i = 2:length(thumbsid_Vals) % Accumulate the eigenvalue sum
thumbsid_CVals(i) = thumbsid_CVals(i-1) + thumbsid_Vals(i);
end;
thumbsid_CVals = thumbsid_CVals / sum(thumbsid_Vals);
```

```
%%//////////////////////////////////////////////////////////////////////
 %plot(upone_Vals);
upone_CVals = zeros(1,length(upone_Vals));
upone_CVals(1) = upone_Vals(1);
for i = 2:length(upone_Vals) % Accumulate the eigenvalue sum
upone_CVals(i) = upone_CVals(i-1) + upone_Vals(i);
end;
upone_CVals = upone_CVals / sum(upone_Vals);
%%//////////////////////////////////////////////////////////////////////
%plot(CVals);
%ylim([0 1]);

%Proj = Vecs(:,1:10)' * Images;
aok = aok_Vecs(:,1:10)' * aok_Images;
bend = bend_Vecs(:,1:10)' * bend_Images;
clos = clos_Vecs(:,1:10)' * clos_Images;
dib = dib_Vecs(:,1:10)' * dib_Images;
dib2n = dib2n_Vecs(:,1:10)' * dib2n_Images;
five = five_Vecs(:,1:10)' * five_Images;
nwcc = nwcc_Vecs(:,1:10)' * nwcc_Images;
ohh = ohh_Vecs(:,1:10)' * ohh_Images;
peace = peace_Vecs(:,1:10)' * peace_Images;
pistol = pistol_Vecs(:,1:10)' * pistol_Images;
power = power_Vecs(:,1:10)' * power_Images;
point = point_Vecs(:,1:10)' * point_Images;
pointL = pointL_Vecs(:,1:10)' *pointL_Images;
punch = punch_Vecs(:,1:10)' * punch_Images;
r = r_Vecs(:,1:10)' *r_Images;
snake = snake_Vecs(:,1:10)' * snake_Images;
swan = swan_Vecs(:,1:10)' * swan_Images;
thumb = thumb_Vecs(:,1:10)' * thumb_Images;
thumbsid = thumbsid_Vecs(:,1:10)' * thumbsid_Images;
upone = upone_Vecs(:,1:10)' * upone_Images;
```

```
hold on
%plot(aok(5,:),aok(2,:),'m*')
%plot(five(5,:),five(2,:),'m<')
%plot(aok(5,:),aok(3,:),'m^')
%plot(aok(5,:),aok(4,:),'mv')
%plot(five(5,:),five(5,:),'ms')
plot(aok(5,:),aok(2,:),'ks')
%plot(five(5,:),five(2,:),'m<')
plot(aok(5,:),aok(3,:),'ks')
plot(aok(5,:),aok(4,:),'ks')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(bend(5,:),bend(2,:),'mv')
%plot(five(5,:),five(2,:),'m<')
plot(bend(5,:),bend(3,:),'mv')
plot(bend(5,:),bend(4,:),'mv')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(clos(5,:),clos(2,:),'r.')
%plot(five(5,:),five(2,:),'m<')
plot(clos(5,:),clos(3,:),'r.')
plot(clos(5,:),clos(4,:),'r.')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(dib(5,:),dib(2,:),'gx')
%plot(five(5,:),five(2,:),'m<')
plot(dib(5,:),dib(3,:),'gx')
plot(dib(5,:),dib(4,:),'gx')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(dib2n(5,:),dib2n(2,:),'b>')
%plot(five(5,:),five(2,:),'m<')
```

```
plot(dib2n(5,:),dib2n(3,:),'b>')
plot(dib2n(5,:),dib2n(4,:),'b>')
%plot(five(5,:),five(5,:),'ms')

%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////

plot(five(5,:),five(2,:),'c*')
%plot(five(5,:),five(2,:),'m<')
plot(five(5,:),five(3,:),'c*')
plot(five(5,:),five(4,:),'c*')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(nwcc(5,:),nwcc(2,:),'r*')
%plot(five(5,:),five(2,:),'m<')
plot(nwcc(5,:),nwcc(3,:),'r*')
plot(nwcc(5,:),nwcc(4,:),'r*')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////

plot(ohh(5,:),ohh(2,:),'b*')
%plot(five(5,:),five(2,:),'m<')
plot(ohh(5,:),ohh(3,:),'b*')
plot(ohh(5,:),ohh(4,:),'b*')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(peace(5,:),peace(2,:),'gs')
%plot(five(5,:),five(2,:),'m<')
plot(peace(5,:),peace(3,:),'gs')
plot(peace(5,:),peace(4,:),'gs')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(pistol(5,:),pistol(2,:),'r+')
```

```
%plot(five(5,:),five(2,:),'m<')
plot(pistol(5,:),pistol(3,:),'r+')
plot(pistol(5,:),pistol(4,:),'r+')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////

%%//////////////////////////////////////////////////////////////////////////
plot(point(5,:),point(2,:),'b.')
%plot(five(5,:),five(2,:),'m<')
plot(point(5,:),point(3,:),'b.')
plot(point(5,:),point(4,:),'b.')
%plot(five(5,:),five(5,:),'ms')

%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////

plot(pointL(5,:),pointL(2,:),'k*')
%plot(five(5,:),five(2,:),'m<')
plot(pointL(5,:),pointL(3,:),'k*')
plot(pointL(5,:),pointL(4,:),'k*')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////
plot(power(5,:),power(2,:),'ms')
%plot(five(5,:),five(2,:),'m<')
plot(power(5,:),power(3,:),'ms')
plot(power(5,:),power(4,:),'ms')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////////

plot(punch(5,:),punch(2,:),'g.')
%plot(five(5,:),five(2,:),'m<')
plot(punch(5,:),punch(3,:),'g.')
```

```
plot(punch(5,:),punch(4,:),'g.')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////
plot(r(5,:),r(2,:),'cs')
%plot(five(5,:),five(2,:),'m<')
plot(r(5,:),r(3,:),'cs')
plot(r(5,:),r(4,:),'cs')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////
plot(snake(5,:),snake(2,:),'k<')
%plot(five(5,:),five(2,:),'m<')
plot(snake(5,:),snake(3,:),'k<')
plot(snake(5,:),snake(4,:),'k<')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////
plot(swan(5,:),swan(2,:),'k.')
%plot(five(5,:),five(2,:),'m<')
plot(swan(5,:),swan(3,:),'k.')
plot(swan(5,:),swan(4,:),'k.')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////
plot(thumb(5,:),thumb(2,:),'b+')
%plot(five(5,:),five(2,:),'m<')
plot(thumb(5,:),thumb(3,:),'b+')
plot(thumb(5,:),thumb(4,:),'b+')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////
plot(thumbsid(5,:),thumbsid(2,:),'m.')
%plot(five(5,:),five(2,:),'m<')
plot(thumbsid(5,:),thumbsid(3,:),'m.')
plot(thumbsid(5,:),thumbsid(4,:),'m.')
%plot(five(5,:),five(5,:),'ms')
%%//////////////////////////////////////////////////////////////////////
```

```
plot(upone(5,:),upone(2,:),'rs')
%plot(five(5,:),five(2,:),'m<')
plot(upone(5,:),upone(3,:),'rs')
plot(upone(5,:),upone(4,:),'rs')
%plot(five(5,:),five(5,:),'ms')
%%////////////////////////////////////////////////////////////////////////////
```

hold off

## .4 GestureFace layer code components

### .4.1 Hand recognition using Opencv library

A simple algorithm for recognising hand gestures has been written. The haar classifier ( mono_20_hand.xml ) trained during this research is utilised.

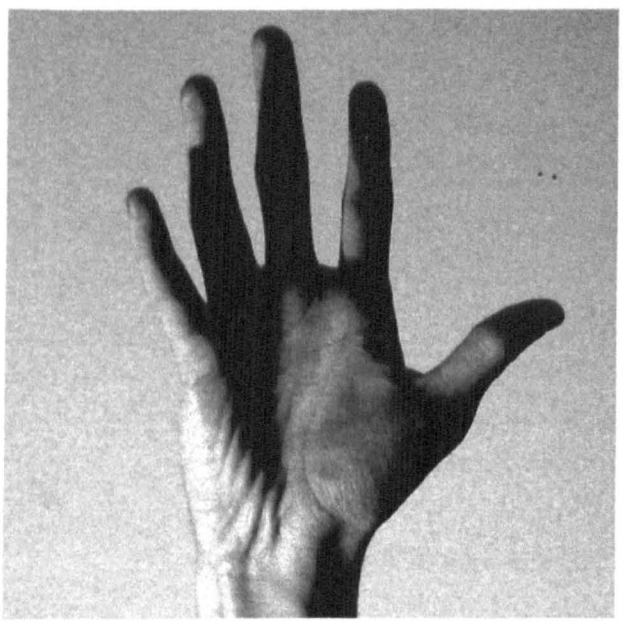


Figure 7: Control posture for Haar classifier mono_20_hand.xml

### .4.1.1 code

```
/*
Control posture Haar classifier mono_20_hand.xml
*/



#include <stdio.h>
#include <OpenCV/cv.h>
#include <OpenCV/highgui.h>
//#include <OpenCV/cvaux.h>

CvHaarClassifierCascade *cascade;
CvMemStorage    *storage;

void detectHand( IplImage *img );

int main( int argc, char** argv )
{
 CvCapture *capture;
 IplImage *frame;
 int key;
 char* filename = "/Users/lupo/Desktop/code_pad/opencv-motion_frameworks/sketches/m



 //char* filename = "/Users/lupo/Desktop/code_pad/opencv-motion_frameworks/sketches,

 cascade = ( CvHaarClassifierCascade* )cvLoad( filename, 0, 0, 0 );
 storage = cvCreateMemStorage( 0 );

// const char *filenameAVI="/Users/lupo/Desktop/code_pad/opencv-motion_frameworks/m
```

```
// const char *videofilename="/Users/lupo/Desktop/code_pad/opencv-motion_frameworks/
// capture = cvCaptureFromFile( argv[1] );
  //capture = cvCreateFileCapture(filenameAVI);

  capture = cvCaptureFromCAM( 0 );

  assert( cascade && storage && capture );

  cvNamedWindow( "video", 1 );

  while( key != 'q' ) {
    frame = cvQueryFrame( capture );

    if( !frame ) {
    // fprintf( stderr, "Cannot query frame!\n" );
      break;
    }

    cvFlip( frame, frame, 1 );
    frame->origin = 0;

    detectHand( frame );

    key = cvWaitKey( 10 );
  }

  cvReleaseCapture( &capture );
  cvDestroyWindow( "video" );
  cvReleaseHaarClassifierCascade( &cascade );
  cvReleaseMemStorage( &storage );

  return 0;
}
```

```
void detectHand( IplImage *img )
{
  int i;

  CvSeq *faces = cvHaarDetectObjects(
                    img,
                    cascade,
                    storage,
                    1.1,
                    3,
                    0 /*CV_HAAR_DO_CANNY_PRUNNING*/,
                    cvSize( 40, 40 ) );

  for( i = 0 ; i < ( faces ? faces->total : 0 ) ; i++ ) {
    CvRect *r = ( CvRect* )cvGetSeqElem( faces, i );
    cvRectangle( img,
          cvPoint( r->x, r->y ),
          cvPoint( r->x + r->width, r->y + r->height ),
          CV_RGB( 255, 0, 0 ), 1, 8, 0 );
  }

  cvShowImage( "video", img );
}
```

## .4.2 Haar Classifier for Hand recognition using Opencv library

In order for the code in section .4.1.1 to function the following haar classifier( mono_20_hand.xml ) trained during this research is needed.

### .4.2.1 code

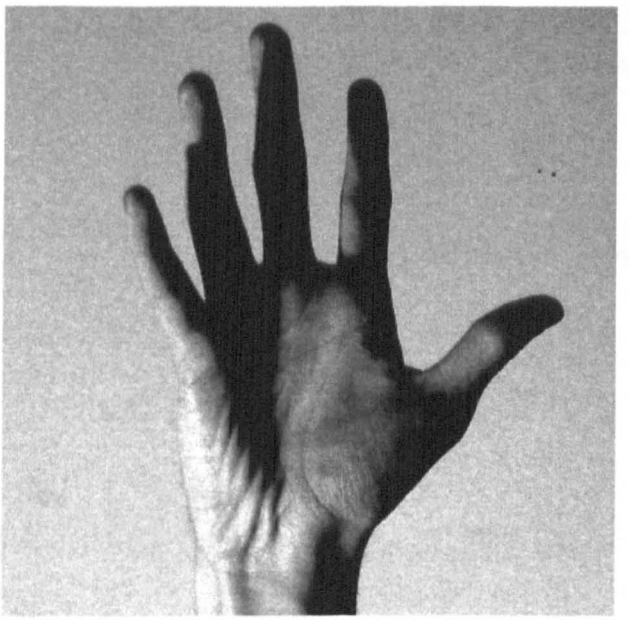


Figure 8: Control posture for Haar classifier mono_20_hand.xml

```
<?xml version="1.0"?>
<opencv_storage>
<mono_20_hand type_id="opencv-haar-classifier">
 <size>
 20 20</size>
 <stages>
  <_>
   <!-- /*
Control posture Haar classifier mono_20_hand.xml
-->
  <!-- stage 0 -->
  <trees>
   <_>
   <!-- tree 0 -->
   <_>
    <!-- root node -->
```

<feature>
<rects>
<_>
2 2 18 18 −1.</_>
<_>
8 2 6 18 3.</_></rects>
<tilted>0</tilted></feature>
<threshold>−0.5153241157531738</threshold>
<left_val>0.9208459854125977</left_val>
<right_val>−0.9449468851089478</right_val></_></_>
<_>
<!−− tree 1 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
8 3 12 5 −1.</_>
<_>
12 7 4 5 3.</_></rects>
<tilted>1</tilted></feature>
<threshold>−0.1828908026218414</threshold>
<left_val>0.7676910758018494</left_val>
<right_val>−0.8145673274993897</right_val></_></_>
<_>
<!−− tree 2 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
4 9 2 6 −1.</_>
<_>

```
   4 9 1 3 2.</_>
   <_>
   5 12 1 3 2.</_></rects>
   <tilted>0</tilted></feature>
  <threshold>-3.4629479050636292e-003</threshold>
  <left_val>0.7557852268218994</left_val>
  <right_val>-0.6670348048210144</right_val></_></_></trees>
 <stage_threshold>-1.0037289857864380</stage_threshold>
 <parent>-1</parent>
 <next>-1</next></_>
<_>
<!-- stage 1 -->
<trees>
 <_>
  <!-- tree 0 -->
  <_>
   <!-- root node -->
   <feature>
    <rects>
     <_>
      1 8 18 12 -1.</_>
     <_>
      7 8 6 12 3.</_></rects>
    <tilted>0</tilted></feature>
   <threshold>-0.3833411931991577</threshold>
   <left_val>0.7563046216964722</left_val>
   <right_val>-0.8750851750373840</right_val></_></_>
 <_>
  <!-- tree 1 -->
  <_>
   <!-- root node -->
   <feature>
    <rects>
```

```
  <_>
   4 5 4 15 -1.</_>
   <_>
   4 10 4 5 3.</_></rects>
  <tilted>0</tilted></feature>
  <threshold>-0.0478372909128666</threshold>
  <left_val>0.6142796874046326</left_val>
  <right_val>-0.8544300198554993</right_val></_></_></trees>
 <stage_threshold>-0.2608054876327515</stage_threshold>
 <parent>0</parent>
 <next>-1</next></_>
<_>
 <!-- stage 2 -->
 <trees>
 <_>
  <!-- tree 0 -->
  <_>
  <!-- root node -->
  <feature>
   <rects>
   <_>
   10 1 6 15 -1.</_>
   <_>
   12 1 2 15 3.</_></rects>
  <tilted>0</tilted></feature>
  <threshold>-0.0461920201778412</threshold>
  <left_val>0.7325975894927979</left_val>
  <right_val>-0.8286684751510620</right_val></_></_>
 <_>
 <!-- tree 1 -->
 <_>
 <!-- root node -->
 <feature>
```

```
<rects>
<_>
4 2 6 18 −1.</_>
<_>
7 2 3 18 2.</_></rects>
<tilted>0</tilted></feature>
<threshold>−0.0848178863525391</threshold>
<left_val>0.5729647874832153</left_val>
<right_val>−0.8650729060173035</right_val></_></_></trees>
<stage_threshold>−0.2557035982608795</stage_threshold>
<parent>1</parent>
<next>−1</next></_>
<_>
<!−− stage 3 −−>
<trees>
<_>
<!−− tree 0 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
5 7 3 4 −1.</_>
<_>
4 8 3 2 2.</_></rects>
<tilted>1</tilted></feature>
<threshold>−0.0132555803284049</threshold>
<left_val>0.7244282960891724</left_val>
<right_val>−0.7451177835464478</right_val></_></_>
<_>
<!−− tree 1 −−>
<_>
<!−− root node −−>
```

<feature>
<rects>
<_>
8 0 7 4 −1.</_>
<_>
8 2 7 2 2.</_></rects>
<tilted>0</tilted></feature>
<threshold>−0.0155559601262212</threshold>
<left_val>0.6760141253471375</left_val>
<right_val>−0.6645848155021668</right_val></_></_>
<_>
<!−− tree 2 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
10 14 1 6 −1.</_>
<_>
10 17 1 3 2.</_></rects>
<tilted>0</tilted></feature>
<threshold>1.4264250239648391e−005</threshold>
<left_val>0.4036096036434174</left_val>
<right_val>−0.9664012789726257</right_val></_></_>
<_>
<!−− tree 3 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
10 14 1 6 −1.</_>
<_>

```
    10 17 1 3 2.</_></rects>
   <tilted>0</tilted></feature>
  <threshold>-1.4289879800344352e-005</threshold>
  <left_val>-1.</left_val>
  <right_val>0.5169605016708374</right_val></_></_></trees>
 <stage_threshold>-0.4891324043273926</stage_threshold>
 <parent>2</parent>
 <next>-1</next></_>
<_>
 <!-- stage 4 -->
 <trees>
 <_>
  <!-- tree 0 -->
  <_>
   <!-- root node -->
   <feature>
   <rects>
    <_>
    2 2 18 18 -1.</_>
    <_>
    8 8 6 6 9.</_></rects>
   <tilted>0</tilted></feature>
  <threshold>-0.8170905113220215</threshold>
  <left_val>0.6477488875389099</left_val>
  <right_val>-0.8245990872383118</right_val></_></_>
 <_>
  <!-- tree 1 -->
  <_>
   <!-- root node -->
   <feature>
   <rects>
    <_>
    9 6 2 14 -1.</_>
```

```
 <_>
  9 13 2 7 2.</_></rects>
 <tilted>0</tilted></feature>
 <threshold>-2.3942769039422274e-003</threshold>
 <left_val>0.4633279144763947</left_val>
 <right_val>-0.8997706174850464</right_val></_></_>
<_>
 <!-- tree 2 -->
 <_>
 <!-- root node -->
 <feature>
 <rects>
  <_>
  9 10 2 9 -1.</_>
  <_>
  9 13 2 3 3.</_></rects>
 <tilted>0</tilted></feature>
 <threshold>-4.3073261622339487e-005</threshold>
 <left_val>-0.9647107720375061</left_val>
 <right_val>0.3608011901378632</right_val></_></_>
<_>
 <!-- tree 3 -->
 <_>
 <!-- root node -->
 <feature>
 <rects>
  <_>
  16 1 1 15 -1.</_>
  <_>
  16 6 1 5 3.</_></rects>
 <tilted>0</tilted></feature>
 <threshold>-0.0122408699244261</threshold>
 <left_val>0.5329871177673340</left_val>
```

<right_val>−0.6351749897003174</right_val></_></_></trees>
<stage_threshold>−0.8305814266204834</stage_threshold>
<parent>3</parent>
<next>−1</next></_>
<_>
<!−− stage 5 −−>
<trees>
<_>
<!−− tree 0 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
7 5 3 6 −1.</_>
<_>
8 5 1 6 3.</_></rects>
<tilted>0</tilted></feature>
<threshold>−0.0101621402427554</threshold>
<left_val>0.6818826198577881</left_val>
<right_val>−0.6540923118591309</right_val></_></_>
<_>
<!−− tree 1 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
9 12 1 2 −1.</_>
<_>
9 12 1 1 2.</_></rects>
<tilted>1</tilted></feature>
<threshold>−6.7364817368797958e−005</threshold>

<left_val>−0.8971663117408752</left_val>
<right_val>0.4335525035858154</right_val></_></_>
<_>
<!−− tree 2 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
2 1 18 18 −1.</_>
<_>
8 1 6 18 3.</_></rects>
<tilted>0</tilted></feature>
<threshold>−0.4840528070926666</threshold>
<left_val>0.3548465967178345</left_val>
<right_val>−0.9229689240455627</right_val></_></_>
<_>
<!−− tree 3 −−>
<_>
<!−− root node −−>
<feature>
<rects>
<_>
10 4 4 6 −1.</_>
<_>
10 4 2 3 2.</_>
<_>
12 7 2 3 2.</_></rects>
<tilted>0</tilted></feature>
<threshold>1.0808430379256606e−003</threshold>
<left_val>−0.8260114192962647</left_val>
<right_val>0.3436168134212494</right_val></_></_></trees>
<stage_threshold>−0.8527951240539551</stage_threshold>

<parent>4</parent>
<next>-1</next></_></stages></mono_20_hand>
</opencv_storage>

## .4.3 Mahalanobis Distancing using Opencv library

This code can be used to calculate the covariance between multiple high or low dimensional vectors.

### .4.3.1 code

```
#include <iostream>

#include <fstream>

#include <cv.h>

#include <highgui.h>

using namespace std;

using namespace cv;

int main( int argc, char** argv )

{

  double distance;
  // Input matrix size

  const int rows = 10;

  const int cols = 6;
  ////////////////////////////////////////////////////////
```

```
 //
 //    Load images into memory
 //
 ////////////////////////////////////////////////////////////

 if (argc==1){
 cout << "No images to load!" << endl;
 cin.get();
 return 0;
}

 int index = 0;
int image_num = argc-1;

Mat *img = new Mat[image_num]; // allocates table on heap instead of stack

 ////////////////////////////////////////////////////////////
 //
// Load the images from command line:
 //
 ////////////////////////////////////////////////////////////
for (index = 0; index < image_num; index++) {
 img[index] = imread(argv[index+1]);
 if (!img[index].data){
  cout << "Image data not loaded properly" << endl;
  cin.get();
  return -1;
 }
}

 for (index = 0; index < image_num; index++) {
 imshow("myWin", img[index]);
   cout << "Image loaded" << endl;
```

```
 waitKey(0);
}
cvDestroyWindow("myWin");

delete [] img; // notice the [] when deleting an array.
// return 0;



 CvArr* newpair;

 newpair = cvCreateMat(1,cols,CV_32FC1);

 cvSetReal1D(newpair, 0, 95);

 cvSetReal1D(newpair, 1, 4);

 cvSetReal1D(newpair, 2, 23);

 cvSetReal1D(newpair, 3, 27);

 cvSetReal1D(newpair, 4, 6);

 cvSetReal1D(newpair, 5, 5);



// Input matrix

float x[rows][cols] = {{95,4,23,27,6,5}, {91,1,20,24,5,5}, {89,1,39,43,16,16},

  {88,3,11,21,4,4},{82,4,6,5,4,3},{90,0,21,23,5,4},{90,0,30,27,9,8},
```

```
  {88,1,24,26,8,7},{90,4,33,30,17,11},{92,2,28,32,8,9}};
// Place input into CvMat**
cout << "\nEnter information in 10x6 matrix" << endl;
CvMat** input = new CvMat*[rows];
for(int i=0; i<rows; i++) {
  input[i] = cvCreateMat(1, cols, CV_32FC1);
  for(int j=0; j<cols; j++) {
    cvmSet(input[i], 0, j, x[i][j]);
  }
}
CvMat* output = cvCreateMat(cols, cols, CV_32FC1);
CvMat* meanvec = cvCreateMat(1, cols, CV_32FC1);
CvMat* inversecovar = cvCreateMat(cols, cols, CV_32FC1);
CvMat* newoutput = cvCreateMat(cols, cols, CV_32FC1);
// Calculate covariance matrix
cout << "\nCalculate covariance matrix" << endl;
cvCalcCovarMatrix((const void **) input, rows, output, meanvec, CV_COVAR_NORMAL);
```

```
// Edit Covar values to match MATLAB & Wolframalpha
cout << "\nPrint out edited covariance matrix" << endl;
for(int i=0; i<cols; i++)
{
  for(int j=0; j<cols; j++)
  {
    cvSetReal2D(newoutput, i, j, cvGetReal2D(output,i,j) / (rows - 1));
    cout << "Edited covariance(" <<i<<","<<j<<"): ";
    printf ("%f\n", cvGetReal2D(newoutput,i,j));
    cout << "\t";
  }
  cout << endl;
}
// To invert and to apply Mahalanobis
cvInvert( newoutput, inversecovar, CV_LU);
distance = cvMahalanobis( meanvec, newpair, inversecovar);
```

```
  printf ("Mahalanobis: %f ", distance);

  // Clear OpenCV datastructures

  cvReleaseMat(&output);

  cvReleaseMat(&meanvec);

  for(int i=0; i<rows; i++)

    cvReleaseMat(&input[i]);

  delete [] input;

  return 0;

}
```

## .4.4 Stereo Disparity using Opencv library

An algorithm for combining two parallel images into a single image map capable of being used for determining depth.

### .4.4.1 code

```
/*
* stereo_match.cpp
* calibration
*

*
*/
```

```
#include <cv.h>
#include <highgui.h>
#include <stdio.h>

using namespace cv;

void saveXYZ(const char* filename, const Mat& mat)
{
 const double max_z = 1.0e4;
 FILE* fp = fopen(filename, "wt");
 for(int y = 0; y < mat.rows; y++)
 {
  for(int x = 0; x < mat.cols; x++)
  {
   Vec3f point = mat.at<Vec3f>(y, x);
   if(fabs(point[2] - max_z) < FLT_EPSILON || fabs(point[2]) > max_z) continue;
   fprintf(fp, "%f %f %f\n", point[0], point[1], point[2]);
  }
 }
 fclose(fp);
}

void print_help()
{
 printf("Usage: stereo_match <left_image> <right_image> [--algorithm=bm|sgbm|hh] [--l
   "[--max-disparity=<max_disparity>] [-i <intrinsic_filename>] [-e <extrinsic_filenam
   "[--no-display] [-o <disparity_image>] [-p <point_cloud_file>]\n");
}

int main(int argc, char** argv)
{
 const char* algorithm_opt = "--algorithm=";
 const char* maxdisp_opt = "--max-disparity=";
```

```
const char* blocksize_opt = "--blocksize=";
const char* nodisplay_opt = "--no-display=";

if(argc < 3)
{
 print_help();
 return 0;
}
const char* img1_filename = 0;
const char* img2_filename = 0;
const char* intrinsic_filename = 0;
const char* extrinsic_filename = 0;
const char* disparity_filename = 0;
const char* point_cloud_filename = 0;

enum { STEREO_BM=0, STEREO_SGBM=1, STEREO_HH=2 };
int alg = STEREO_SGBM;
int SADWindowSize = 0, numberOfDisparities = 0;
bool no_display = false;

StereoBM bm;
StereoSGBM sgbm;

for( int i = 1; i < argc; i++ )
{
 if( argv[i][0] != '-' )
 {
  if( !img1_filename )
   img1_filename = argv[i];
  else
   img2_filename = argv[i];
 }
 else if( strncmp(argv[i], algorithm_opt, strlen(algorithm_opt)) == 0 )
```

```
{
 char* _alg = argv[i] + strlen(algorithm_opt);
 alg = strcmp(_alg, "bm") == 0 ? STEREO_BM :
    strcmp(_alg, "sgbm") == 0 ? STEREO_SGBM :
    strcmp(_alg, "hh") == 0 ? STEREO_HH : -1;
 if( alg < 0 )
 {
  printf("Command-line parameter error: Unknown stereo algorithm\n\n");
  print_help();
  return -1;
 }
}
else if( strncmp(argv[i], maxdisp_opt, strlen(maxdisp_opt)) == 0 )
{
 if( sscanf( argv[i] + strlen(maxdisp_opt), "%d", &numberOfDisparities ) != 1 ||
     numberOfDisparities < 1 || numberOfDisparities % 16 != 0 )
 {
  printf("Command-line parameter error: The max disparity (--maxdisparity=<...>) mu
  print_help();
  return -1;
 }
}
else if( strncmp(argv[i], blocksize_opt, strlen(blocksize_opt)) == 0 )
{
 if( sscanf( argv[i] + strlen(blocksize_opt), "%d", &SADWindowSize ) != 1 ||
     SADWindowSize < 1 || SADWindowSize % 2 != 1 )
 {
  printf("Command-line parameter error: The block size (--blocksize=<...>) must be a
  return -1;
 }
}
else if( strcmp(argv[i], nodisplay_opt) == 0 )
 no_display = true;
```

```
 else if( strcmp(argv[i], "-i" ) == 0 )
  intrinsic_filename = argv[++i];
 else if( strcmp(argv[i], "-e" ) == 0 )
  extrinsic_filename = argv[++i];
 else if( strcmp(argv[i], "-o" ) == 0 )
  disparity_filename = argv[++i];
 else if( strcmp(argv[i], "-p" ) == 0 )
  point_cloud_filename = argv[++i];
 else
 {
  printf("Command-line parameter error: unknown option %s\n", argv[i]);
  return -1;
 }
}

if( !img1_filename || !img2_filename )
{
 printf("Command-line parameter error: both left and right images must be specified\n");
 return -1;
}

if( (intrinsic_filename != 0) ^ (extrinsic_filename != 0) )
{
 printf("Command-line parameter error: either both intrinsic and extrinsic parameters mus
 return -1;
}

if( extrinsic_filename == 0 && point_cloud_filename )
{
 printf("Command-line parameter error: extrinsic and intrinsic parameters must be specific
 return -1;
}
```

```
int color_mode = alg == STEREO_BM ? 0 : -1;
Mat img1 = imread(img1_filename, color_mode);
Mat img2 = imread(img2_filename, color_mode);
Size img_size = img1.size();

Rect roi1, roi2;
Mat Q;

if( intrinsic_filename )
{
 // reading intrinsic parameters
 FileStorage fs(intrinsic_filename, CV_STORAGE_READ);
 if(!fs.isOpened())
 {
  printf("Failed to open file %s\n", intrinsic_filename);
  return -1;
 }

 Mat M1, D1, M2, D2;
 fs["M1"] >> M1;
 fs["D1"] >> D1;
 fs["M2"] >> M2;
 fs["D2"] >> D2;

 fs.open(extrinsic_filename, CV_STORAGE_READ);
 if(!fs.isOpened())
 {
  printf("Failed to open file %s\n", extrinsic_filename);
  return -1;
 }

 Mat R, T, R1, P1, R2, P2;
 fs["R"] >> R;
```

```
fs["T"] >> T;

stereoRectify( M1, D1, M2, D2, img_size, R, T, R1, R2, P1, P2, Q, -1, img_size, &roi1, &r

Mat map11, map12, map21, map22;
initUndistortRectifyMap(M1, D1, R1, P1, img_size, CV_16SC2, map11, map12);
initUndistortRectifyMap(M2, D2, R2, P2, img_size, CV_16SC2, map21, map22);

Mat img1r, img2r;
remap(img1, img1r, map11, map12, INTER_LINEAR);
remap(img2, img2r, map21, map22, INTER_LINEAR);

img1 = img1r;
img2 = img2r;
}

numberOfDisparities = numberOfDisparities > 0 ? numberOfDisparities : img_size.width/8;

bm.state->roi1 = roi1;
bm.state->roi2 = roi2;
bm.state->preFilterCap = 31;
bm.state->SADWindowSize = SADWindowSize > 0 ? SADWindowSize : 9;
bm.state->minDisparity = 0;
bm.state->numberOfDisparities = numberOfDisparities;
bm.state->textureThreshold = 10;
bm.state->uniquenessRatio = 15;
bm.state->speckleWindowSize = 100;
bm.state->speckleRange = 32;
bm.state->disp12MaxDiff = 1;

sgbm.preFilterCap = 63;
sgbm.SADWindowSize = SADWindowSize > 0 ? SADWindowSize : 3;
```

```
int cn = img1.channels();

sgbm.P1 = 8*cn*sgbm.SADWindowSize*sgbm.SADWindowSize;
sgbm.P2 = 32*cn*sgbm.SADWindowSize*sgbm.SADWindowSize;
sgbm.minDisparity = 0;
sgbm.numberOfDisparities = numberOfDisparities;
sgbm.uniquenessRatio = 10;
sgbm.speckleWindowSize = bm.state->speckleWindowSize;
sgbm.speckleRange = bm.state->speckleRange;
sgbm.disp12MaxDiff = 1;
sgbm.fullDP = alg == STEREO_HH;

Mat disp, disp8;
//Mat img1p, img2p, dispp;
//copyMakeBorder(img1, img1p, 0, 0, numberOfDisparities, 0, IPL_BORDER_REPLICATE
//copyMakeBorder(img2, img2p, 0, 0, numberOfDisparities, 0, IPL_BORDER_REPLICATE

int64 t = getTickCount();
if( alg == STEREO_BM )
 bm(img1, img2, disp);
else
 sgbm(img1, img2, disp);
t = getTickCount() - t;
printf("Time elapsed: %fms\n", t*1000/getTickFrequency());

//disp = dispp.colRange(numberOfDisparities, img1p.cols);
disp.convertTo(disp8, CV_8U, 255/(numberOfDisparities*16.));
if( !no_display )
{
 namedWindow("left", 1);
 imshow("left", img1);
 namedWindow("right", 1);
 imshow("right", img2);
```

```
  namedWindow("disparity", 0);
  imshow("disparity", disp8);
  printf("press any key to continue...");
  fflush(stdout);
  waitKey();
  printf("\n");
 }

 if(disparity_filename)
  imwrite(disparity_filename, disp8);

 if(point_cloud_filename)
 {
  printf("storing the point cloud...");
  fflush(stdout);
  Mat xyz;
  reprojectImageTo3D(disp, xyz, Q, true);
  saveXYZ(point_cloud_filename, xyz);
  printf("\n");
 }

 return 0;
}
```

## .4.5 Motion History using Opencv library

A method of segmenting human activities from complex backgrounds.

### .4.5.1 code

```
/*
*
* Motion tracking
*
```

```
* Created by Leon Barker on 11/03/2010.
* Copyright 2010 LAB. All rights reserved.
*
*/

#ifdef _CH_
#pragma package <opencv>
#endif

#define CV_NO_BACKWARD_COMPATIBILITY

#ifndef _EiC
// motion templates sample code
#include <OpenCV/cv.h>
#include <OpenCV/highgui.h>
#include <time.h>
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#endif

// various tracking parameters (in seconds)
const double MHI_DURATION = 1;
const double MAX_TIME_DELTA = 0.5;
const double MIN_TIME_DELTA = 0.05;
const int N = 4; // number of cyclic frame buffer used for motion detection // (should, prob
int g_thresh = 100;

IplImage **buf = 0;// ring image buffer
int last = 0;
int isColor = 1;
int fps = 25;
```

```
// temporary images
IplImage* mhi = 0; // MHI
IplImage* orient = 0; // orientation
IplImage* mask = 0; // valid orientation mask
IplImage* segmask = 0; // motion segmentation map
IplImage* motion = 0;
IplImage* image = 0;
CvVideoWriter *writer = 0; //writing video to file
CvMemStorage* g_storage = 0;
CvMemStorage* storage = 0; // temporary storage


//Text Initiation
// Text variables*****************************
const char* text = "Right";
double hscale = 1.0;
double vscale = 0.8;
double shear = 0.2;
int thickness2 = 1;
int line_type = 8;
CvPoint pt2 = cvPoint(205,195);
CvScalar blue = CV_RGB(0,0,250);
CvScalar white = CV_RGB(255,255,255);
// Text variables*****************************


void update_mhi( IplImage* img, IplImage* dst)
{
 double timestamp = (double)clock()/CLOCKS_PER_SEC; // get current time in seconds
 CvSize size = cvSize(img->width,img->height); // get current frame size
 int i, idx1 = last, idx2;
 IplImage* silh;
 CvSeq* seq;
```

```
// allocate images at the beginning or
// reallocate them if the frame size is changed
if( !mhi || mhi->width != size.width || mhi->height != size.height ) {
 if( buf == 0 ) {
  buf = (IplImage**)malloc(N*sizeof(buf[0]));
  memset( buf, 0, N*sizeof(buf[0]));
 }

 for( i = 0; i < N; i++ ) {
  cvReleaseImage( &buf[i] );
  buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
  cvZero( buf[i] );
 }
 cvReleaseImage( &mhi );
 cvReleaseImage( &orient );
 cvReleaseImage( &segmask );
 cvReleaseImage( &mask );

 mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );
 cvZero( mhi ); // clear MHI at the beginning
 orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );
 segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
 mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
}

cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to grayscale
idx2 = (last + 1) % N; // index of (last - (N-1))th frame
last = idx2;
silh = buf[idx2];
cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference between frames
cvThreshold( silh, silh, g_thresh, 1, CV_THRESH_BINARY ); // and threshold it
cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); // update MHI
```

```
// convert MHI to blue 8u image
cvCvtScale( mhi, mask, 255./MHI_DURATION,(MHI_DURATION - timestamp)*255./MH
cvZero( dst );
cvMerge( 0, 0, mask, 0, dst );
// calculate motion gradient orientation and valid orientation mask
cvCalcMotionGradient( mhi, mask, orient, MAX_TIME_DELTA, MIN_TIME_DELTA, 3 );

if( !storage )
 storage = cvCreateMemStorage(0);
else
 cvClearMemStorage(storage);

// segment motion: get sequence of motion components
// segmask is marked motion components map. It is not used further
seq = cvSegmentMotion( mhi, segmask, storage, timestamp, MAX_TIME_DELTA );


}

void on_trackbar(int){



  if( g_storage == NULL ){
    g_storage = cvCreateMemStorage(0);
  } else {
    cvClearMemStorage( g_storage );
  }
  update_mhi( image, motion );

}
```

```
int main(int argc, char** argv)
{
  //*****************************************
  CvFont font1;
  cvInitFont(&font1,CV_FONT_HERSHEY_DUPLEX,hscale,vscale,shear,thickness2,line_type
  //*****************************************


 CvCapture* capture = 0;
  // const char *filename="/Users/lupo/Desktop/code_pad/opencv-motion_frameworks/mo
  // const char *videofilename="/Users/lupo/Desktop/code_pad/opencv-motion_framework
  //capture = cvCaptureFromFile( argv[1] );
  //capture = cvCreateFileCapture(filename);

  if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0]))){
    // capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
    // capture3 = cvCaptureFromCAM(0);
    // capture2 = cvCaptureFromCAM(1);
    capture = cvCaptureFromCAM(0);
 }else if( argc == 2 )
    // capture = cvCaptureFromFile( argv[1] );


    fps = (int)cvGetCaptureProperty( capture, CV_CAP_PROP_FPS );


if( capture )
{

 cvNamedWindow( "Motion", 1 );
    cvCreateTrackbar( "Threshold", "Motion", &g_thresh, 255, on_trackbar );
    image = cvQueryFrame(capture);
    // writer = cvCreateVideoWriter(videofilename,CV_FOURCC('P', 'I', 'M', '1'),fps, cvSiz
```

```
while((image = cvQueryFrame(capture)) != NULL){

    if( !image )
  break;

 if( !motion )
 {
  motion = cvCreateImage( cvSize(image->width,image->height), 8, 3 );
  cvZero( motion );
  motion->origin = image->origin;
 }

    on_trackbar(0);

  /* cvRectangle( motion,
          cvPoint( 450,310 ),
          cvPoint( 160, 150 ),
          CV_RGB( 0, 255, 0 ), 3, 8, 0 );

    cvPutText(motion,text,pt2,&font1,white);*/ //Graphic overlay

    cvShowImage( "Motion", motion );
    cvWriteFrame(writer,motion);

 if( cvWaitKey(10) >= 0 )
  break;
}
  cvReleaseVideoWriter(&writer);
cvReleaseCapture( &capture );
cvDestroyWindow( "Motion" );
}
```

```
 return 0;
}

#ifdef _EiC

#endif
```

# Bibliography


Anton-Canalis, L., Nielsen, E. S. and Castrillon-Santana, M. [2005], Hand pose detection for vision-based gesture interfaces, *in* 'Proceedings of the 2005 Conference on Machine Vision Applications', Japan, pp. 13 17. 67, 69

Armstrong, T. J., Foulke, J. A., Martin, B. J., Gerson, J. and Rempel, D. M. [1994], 'Investigation of applied forces in alphanumeric keyboard work', *Am Ind Hyg Assoc J* **55**(1), 30 35.
**URL:** *http://www.hubmed.org/display.cgi?uids=8116526* 41

Billinghurst, M. and Kato, H. [2002], 'Collaborative augmented reality', *COMMUNICATIONS OF THE ACM* **45**, 64 70. 7

Birchfield, S. T., Birchfield, S. T., Tomasi, C. and Fraser-smith, A. [1999], 'Depth and motion discontinuities'. xv, 54, 63, 64, 65

Bobick, A. F., Davis, J. W., Society, I. C. and Society, I. C. [2001], 'The recognition of human movement using temporal templates', *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* **23**, 257 267. 60

Bolt, R. A. [1980], '"put-that-there" voice and gesture at the graphics interface', *SIGGRAPH Comput. Graph.* **14**, 262 270.
**URL:** *http://doi.acm.org/10.1145/965105.807503* xii, 7

Bowden, R., Zisserman, A., Kadir, T. and Brady, M. [2003], 'Vision based interpretation of natural sign languages', *IN: EXHIBITION AT ICVS03: THE*

*3RD INTERNATIONAL CONFERENCE ON COMPUTER VISION SYSTEMS* pp. 391–401. xii, 2, 15, 16, 33

Cadoz, C. [1998], *Die virtuelle Realität*, dt. erstveröff. edn, BLT, Bergisch Gladbach. 36

Coblentz, A. M. [1989], *Vigilance and performance in automatized systems = Vigilance et performance de l'homme dans les systèmes automatisés*, Kluwer Academic Publishers, Dordrecht ;;Boston. 46

Cudd, P., Whiteside, S., Stoneham, H., Syder, D. and De Bruijn, C. [1998], Using dictation systems: A contributory cause of dysphonia, *in* 'VoiceData98', pp. 98–103. 39

Daveluy, C. [2000], *Enquête sociale et de santé 1998.*, Institut de la statistique du Québec, [Québec]. 43

David, P. A. [1985], 'Clio and the economics of qwerty', *The American Economic Review* **75**(2), pp. 332–337.
**URL:** *http://www.jstor.org/stable/1805621* 125

Delisle, A., Larivière, C., Imbeau, D. and Durand, M.-J. [2005], 'Physical exposure of sign language interpreters: baseline measures and reliability analysis', *European Journal of Applied Physiology* **94**(4), 448–460. 43

Efron, D. [1941], *Gesture and environment a tenative study of some of the spatio -temporal and "linguistic" aspects of the gestural behavior of eastern Jews and southern Italians in New York city, living under similar*, King's crown press, New York. 36

Fails, J. A. and Jr., D. O. [2002], Light widgets: interacting in every-day spaces, *in* 'Proceedings of the 7th international conference on Intelligent user interfaces', IUI '02, ACM, New York, NY, USA, pp. 63–69.
**URL:** *http://doi.acm.org/10.1145/502716.502729* xii, 14, 15, 73

Farrugia, J. P., Horain, P., Guehenneux, E. and Alusse, Y. [2006], Gpucv: A framework for image processing acceleration with graphics processors, *in* '2006

IEEE International Conference on Multimedia and Expo', IEEE, pp. 585–588. 129

Freeman, W. T., Freeman, W. T. and Weissman, C. D. [1995], 'Television control by hand gestures', *INTERNATIONAL WORKSHOP ON AUTOMATIC FACE AND GESTURE RECOGNITION* pp. 179–183. xii, 10, 11, 24

Gandy, M., Starner, T., Auxier, J. and Ashbrook, D. [2000], The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring, *in* 'Proceedings of the 4th IEEE International Symposium on Wearable Computers', ISWC '00, IEEE Computer Society, Washington, DC, USA, pp. 87–.
**URL:** *http://portal.acm.org/citation.cfm?id=851037.856538* 122

Han, J. Y. [2005*a*], Low-cost multi-touch sensing through frustrated total internal reflection, *in* 'Proceedings of the 18th annual ACM symposium on User interface software and technology', UIST '05, ACM, New York, NY, USA, pp. 115–118.
**URL:** *http://doi.acm.org/10.1145/1095034.1095054* 30

Han, J. Y. [2005*b*], Multi-touch sensing through frustrated total internal reflection, *in* 'ACM SIGGRAPH 2005 Sketches', SIGGRAPH '05, ACM, New York, NY, USA.
**URL:** *http://doi.acm.org/10.1145/1187112.1187287* xii, 16, 17, 18, 19, 121

Hauptmann, A. G. and McAvinney, P. [1993], 'Gestures with speech for graphic manipulation', *Int. J. Man-Mach. Stud.* **38**, 231–249.
**URL:** *http://portal.acm.org/citation.cfm?id=160032.160036* 8, 39

Iacoboni, M., Woods, R., Brass, M., Bekkering, H., Mazziotta, J. and Rizzolatti, G. [1999], 'Cortical mechanisms of human imitation', *Science* **286**(5449), 2526–2528. 40

Iannizzotto, G., Villari, M. and Vita, L. [2001], Hand tracking for human-computer interaction with graylevel visualglove: turning back to the simple way, *in* 'Proceedings of the 2001 workshop on Perceptive user interfaces', PUI

'01, ACM, New York, NY, USA, pp. 1–7.
**URL:** *http://doi.acm.org/10.1145/971478.971512* xii, 13

Jeffries, R. and Desurvire, H. [1992], 'Usability testing vs. heuristic evaluation: was there a contest?', *SIGCHI Bull.* **24**, 39–41.
**URL:** *http://doi.acm.org/10.1145/142167.142179* 47, 48

Kendon, A. [1994], *Human Gesture: Tools, language, and cognition in human evolution*, 1st paperback ed. edn, Cambridge University Press, Cambridge [England] ;;New York, chapter 1, pp. 43–62. 36

Kortenkamp, D., Huber, E. and Bonasso, R. P. [1996], Recognizing and interpreting gestures on a mobile robot, *in* 'Proceedings of the thirteenth national conference on Artificial intelligence - Volume 2', AAAI'96, AAAI Press, pp. 915–921.
**URL:** *http://portal.acm.org/citation.cfm?id=1864519.1864523* xii, 2, 12

Krueger, M. W., Gionfriddo, T. and Hinrichsen, K. [1985], Videoplace—an artificial reality, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', CHI '85, ACM, New York, NY, USA, pp. 35–40.
**URL:** *http://doi.acm.org/10.1145/317456.317463* xii, 3, 6, 116, 120

Liebowitz, S. J. and Margolis, S. E. [1995], 'Path dependence, lock-in, and history', *Journal of Law, Economics and Organization* **11**(1), 205–26.
**URL:** *http://econpapers.repec.org/RePEc:oup:jleorg:v:11:y:1995:i:1:p:205-26* 125

Lin, J., Wu, Y. and Huang, T. S. [2000], Modeling the constraints of human hand motion, *in* 'Proceedings of the Workshop on Human Motion (HUMO'00)', HUMO '00, IEEE Computer Society, Washington, DC, USA, pp. 121–.
**URL:** *http://portal.acm.org/citation.cfm?id=822088.823446* xii, 14

Marklin, R. W., Simoneau, G. G. and Monroe, J. F. [1999], 'Wrist and forearm posture from typing on split and vertically inclined computer keyboards', *Hum Factors* **41**(4), 559–569.
**URL:** *http://www.hubmed.org/display.cgi?uids=10774127* 41

Marras, W. S. and Schoenmarklin, R. W. [1993], 'Wrist motions in industry', *Ergonomics* **36**(4), 341–351.
**URL:** *http://www.hubmed.org/display.cgi?uids=8472684* 41, 44

Mcneill, D. [1992], *Hand and Mind: What Gestures Reveal about Thought*, University Of Chicago Press. 36, 37

Milgram, P. and Kishino, F. [1994], 'A taxonomy of mixed reality visual displays', *IEICE Transactions on Information Systems* **E77-D**(12). 4, 9

Mistry, P. and Maes, P. [2009], Sixthsense: a wearable gestural interface, *in* 'ACM SIGGRAPH ASIA 2009 Sketches', SIGGRAPH ASIA '09, ACM, New York, NY, USA, pp. 11:1–11:1.
**URL:** *http://doi.acm.org/10.1145/1667146.1667160* 122

Moore, A., Wells, R. and Ranney, D. [1991], 'Quantifying exposure in occupational manual tasks with cumulative trauma disorder potential', *Ergonomics* **34**(12), 1433–1453.
**URL:** *http://www.hubmed.org/display.cgi?uids=1800109* 41

Mulder, A. [1996], Hand gestures for hci: research on human movement behaviour reviewed in the context of hand centred input, Technical report, Simon Fraser University, Canada, http://www.xspasm.com/x/sfu/vmi/HCI-gestures.htm. 36

Munoz-Salinas, R., Medina-Carnicer, R., Madrid-Cuevas, F. J. and Carmona-Poyato, A. [2008], 'Depth silhouettes for gesture recognition', *Pattern Recogn. Lett.* **29**, 319–329.
**URL:** *http://portal.acm.org/citation.cfm?id=1326360.1326445* 66

Neumann, J. and Aloimonos, Y. [2000], Talking heads: Introducing the tool of 3d motion fields in the study of action, *in* 'Proceedings of the Workshop on Human Motion (HUMO'00)', HUMO '00, IEEE Computer Society, Washington, DC, USA, pp. 25–.
**URL:** *http://portal.acm.org/citation.cfm?id=822088.823439* 59

Nielsen, J. [1994], Usability inspection methods, *in* 'Conference companion on Human factors in computing systems', CHI '94, ACM, New York, NY, USA, pp. 413–414.
**URL:** *http://doi.acm.org/10.1145/259963.260531* xii, 45, 46

Nielsen, M., Störring, M., Moeslund, T. B. and Granum, E. [2004], 'A procedure for developing intuitive and ergonomic gesture interfaces for hci'. 31, 45

Norman, D. A. and Fisher, D. [1982], 'Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter', *Human Factors: The Journal of the Human Factors and Ergonomics Society* **11**, 509–519.
**URL:** *http://www.ingentaconnect.com/content/hfes/hf/1982/00000024/00000005/art00002*
125

P, T., S, B. and D., R. [1999], 'Risk factors for musculoskeletal disorders among computer users.', *Occupational medicine (Philadelphia)* **14**(1), 17–38. 41

Pavlovic, V. I., Sharma, R. and Huang, T. S. [1997], 'Visual interpretation of hand gestures for human-computer interaction: A review', *IEEE Trans. Pattern Anal. Mach. Intell.* **19**, 677–695.
**URL:** *http://dx.doi.org/10.1109/34.598226* 31, 38, 39

Premaratne, P. and Nguyen, Q. [2007], 'Consumer electronics control system based on hand gesture moment invariants', *Faculty of Informatics - Papers* . xii, 11

Quek, F., McNeill, D., Bryll, R., Duncan, S., Ma, X.-F., Kirbas, C., McCullough, K. E. and Ansari, R. [2002], 'Multimodal human discourse: gesture and speech', *ACM Trans. Comput.-Hum. Interact.* **9**, 171–193.
**URL:** *http://doi.acm.org/10.1145/568513.568514* 33, 38, 39

Rasmussen, J. [1987], *The definition of human error and a taxonomy for technical system design*, John Wiley and Sons. 46

Rekimoto, J. and Nagao, K. [1995], The world through the computer: computer augmented interaction with real world environments, *in* 'Proceedings of the 8th annual ACM symposium on User interface and software technology', UIST

'95, ACM, New York, NY, USA, pp. 29–36.
**URL:** *http://doi.acm.org/10.1145/215585.215639* xii, 3

Rime, B. and Schiaratura, L. [1991], 'Gesture and speech', pp. 239–281. 36

Rizzolatti, G. and Craighero, L. [2004], 'The mirror-neuron system', *Annual Review of Neuroscience* **27**(1), 169–192.
**URL:** *http://www.annualreviews.org/doi/abs/10.1146/annurev.neuro.27.070203.144230* 40

Rizzolatti, G., Fogassi, L. and Gallese, V. [2001], 'Neurophysiological mechanisms underlying the understanding and imitation of action.', *Nat Rev Neurosci* **2**(9), 661–670. 40

Scheuerle, J., Guilford, A. M. and Habal, M. B. [2000], 'Work-related cumulative trauma disorders and interpreters for the deaf', *Appl Occup Environ Hyg* **15**(5), 429–434.
**URL:** *http://www.hubmed.org/display.cgi?uids=10808265* 43

Schieber, M. H. [1991], 'Individuated finger movements of rhesus monkeys: a means of quantifying the independence of the digits', *J Neurophysiol* **65**(6), 1381–1391.
**URL:** *http://www.hubmed.org/display.cgi?uids=1875247* 41, 88

Schieber, M. H. [1995], 'Muscular production of individuated finger movements: the roles of extrinsic finger muscles', *J Neurosci* **15**(1 Pt 1), 284–297.
**URL:** *http://www.hubmed.org/display.cgi?uids=7823134* 41

Serina, E. R., Tal, R. and Rempel, D. [1999], 'Wrist and forearm postures and motions during typing', *Ergonomics* **42**(7), 938–951.
**URL:** *http://www.hubmed.org/display.cgi?uids=10424183* 42

Shealy, J., Feuerstein, M. and Latko, W. [1991], 'Biomechanical analysis of upper extremity risk in sign langauge interpreting', *Journal of Occupational Rehabilitation* **1**(3), 217–225. 44

Sommerich, C. M., Marras, W. S. and Parnianpour, M. [1996], 'A quantitative description of typing biomechanics', *Journal of Occupational Rehabilitation* **6**(1), 33–55. 42, 43, 75

Starner, T. and Pentland, A. [1995], Real-time american sign language recognition from video using hidden markov models, *in* 'Proceedings of the International Symposium on Computer Vision', IEEE Computer Society, Washington, DC, USA, pp. 265–.
**URL:** *http://portal.acm.org/citation.cfm?id=525981.849918* 2, 33

Stern, H. I., Wachs, J. P. and Edan, Y. [2006], Optimal hand gesture vocabulary design using psycho-physiological and technical factors, *in* 'Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition', FGR '06, IEEE Computer Society, Washington, DC, USA, pp. 257–262.
**URL:** *http://dx.doi.org/10.1109/FGR.2006.84* 117, 120

Stokoe, W. [1960], 'Sign language structure: An outline of the visual communication systems of the american deaf', *Journal of Deaf Studies and Deaf Education* **10**(1), 3. 15, 36

Sutherland, I. E. [1964], Sketch pad a man-machine graphical communication system, *in* 'Proceedings of the SHARE design automation workshop', DAC '64, ACM, New York, NY, USA, pp. 6.329–6.346.
**URL:** *http://doi.acm.org/10.1145/800265.810742* 5

Sutherland, I. E. [1968], A head-mounted three dimensional display, *in* 'Proceedings of the December 9-11, 1968, fall joint computer conference, part I', AFIPS '68 (Fall, part I), ACM, New York, NY, USA, pp. 757–764.
**URL:** *http://doi.acm.org/10.1145/1476589.1476686* 3

Tamura, H. and Yamamoto, H. [1998], Vision and graphics in producing mixed reality worlds, *in* 'Proceedings of the 1998 Workshop on Computer Vision for Virtual Reality Based Human Communications (CVVRHC '98)', IEEE Computer Society, Washington, DC, USA, pp. 0078–.
**URL:** *http://portal.acm.org/citation.cfm?id=551005.795391* 4

Vilkman, E. [2000], 'Voice problems at work: A challenge for occupational safety and health arrangement.', *Folia Phoniatrica Logopaedica* **52**, 120–125. 39

Viney, R. and Green, R. [2007], Gpu-accelerated computer vision on the linux platform, Pro- ceedings of Image and Vision Computing New Zealand, pp. 143–147. 129

Viola, P. and Jones, M. [2001], 'Rapid object detection using a boosted cascade of simple features', *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* **1**, 511. 66

Wachs, J. P., Stern, H. I., Edan, Y., Gillam, M., Handler, J., Feied, C. and Smith, M. [2008], 'A gesture-based tool for sterile browsing of radiology images', *Journal of the American Medical Informatics Association* **15**(3), 321 – 323.
**URL:** *http://www.sciencedirect.com/science/article/B7CPS-4SBHSBM-F/2/f47441f33cddd33d6aeaa860ec752102* 2

Wachs, J., Stern, H., Edan, Y., Gillam, M., Feied, C., Smith, M. and Handler, J. [2007], 'Gestix: A doctor-computer sterile gesture interface for dynamic environments'. xii, 19, 20

Weinland, D., Ronfard, R. and Boyer, E. [2006], 'Free viewpoint action recognition using motion history volumes', *Computer Vision and Image Understanding* **104**(2-3), 249 – 257. Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour.
**URL:** *http://www.sciencedirect.com/science/article/B6WCX-4M21T79-1/2/ca360944405577c574bd94569c506385* 60

Wellner, P. [1991], The digitaldesk calculator: tangible manipulation on a desk top display, *in* 'Proceedings of the 4th annual ACM symposium on User interface software and technology', UIST '91, ACM, New York, NY, USA, pp. 27–33.
**URL:** *http://doi.acm.org/10.1145/120782.120785* xii, 9, 30

Wells, R., Moore, A., Potvin, J. and Norman, R. [1994], 'Assessment of risk factors for development of work-related musculoskeletal disorders (rsi)',

*Applied Ergonomics* **25**(3), 157 – 164.
**URL:** *http://www.sciencedirect.com/science/article/B6V1W-47YDV08-NY/2/ba11a0dc6a5a68101d806f73184747a6* 42

Westerman, W. [1999], Hand tracking, finger identification, and chordic manipulation on a multi-touch surface, PhD thesis, University of Delaware. 30, 33

Wexelblat, A. [1995], 'An approach to natural gesture in virtual environments', *ACM Trans. Comput.-Hum. Interact.* **2**, 179–200.
**URL:** *http://doi.acm.org/10.1145/210079.210080* 33, 38, 39

Wexelblat, A. [1998], 'Research challenges in gesture: Open issues and unsolved problems'. xv, 38

Wiener, E. L. [1988], *Human Factors in Aviation.*, Academic Press Inc., San Diego. 46

Wilson, A. D. [2006], Robust computer vision-based detection of pinching for one and two-handed gesture input, *in* 'Proceedings of the 19th annual ACM symposium on User interface software and technology', UIST '06, ACM, New York, NY, USA, pp. 255–258.
**URL:** *http://doi.acm.org/10.1145/1166253.1166292* xii, 18

Woods, D., Roth, E. and Jr, H. P. [1988], 'Modeling human intention formation for human reliability assessment', *Reliability Engineering System Safety* **22**(1-4), 169 – 200.
**URL:** *http://www.sciencedirect.com/science/article/B6V4T-482B3FV-R/2/bc5ff4663d219cb307903402e388a1fd* 46

Zahedi, M., Keysers, D., Deselaers, T. and Ney, H. [2005], 'Combination of tangent distance and an image distortion model for appearance-based sign language recognition'. 2, 33

Zhai, S., Milgram, P. and Buxton, W. [1996], The influence of muscle groups on performance of multiple degree-of-freedom input, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems: common ground',

CHI '96, ACM, New York, NY, USA, pp. 308–315.
**URL:** *http://doi.acm.org/10.1145/238386.238534* 2

Zhang, D., Li, S. Z. and Gatica-Perez, D. [2004], 'Real-time face detection using boosting in hierarchical feature spaces', *Pattern Recognition, International Conference on* **2**, 411–414. 70

Zieren, J. and Kraiss, K.-F. [2005], 'Robust person-independent visual sign language recognition'. 2, 33